TRANSPORTATION SCIENCE
Vol. 39, No. 1, February 2005, pp. 104–118
ISSN 0041-1655 | EISSN 1526-5447 | 05 | 3901 | 0104

informs®

DOI 10.1287/trsc.1030.0056

Vehicle Routing Problem with Time Windows, Part I: Route Construction and Local Search Algorithms

Olli Bräysy
Agora Innoroad Laboratory, University of Jyväskylä, P. O. Box 35, FIN-40014 Jyväskylä, Finland
olli.braysy@jyu.fi

Michel Gendreau
Département d'informatique et de recherche opérationelle, and Centre de recherche sur les transports, Université de Montréal, C.P. 6128, Succursale Centre-ville, Montréal, Canada H3C 3J7,
michelg@crt.umontreal.ca

This paper presents a survey of the research on the vehicle routing problem with time windows (VRPTW). The VRPTW can be described as the problem of designing least cost routes from one depot to a set of geographically scattered points. The routes must be designed in such a way that each point is visited only once by exactly one vehicle within a given time interval, all routes start and end at the depot, and the total demands of all points on one particular route must not exceed the capacity of the vehicle. Both traditional heuristic route construction methods and recent local search algorithms are examined. The basic features of each method are described, and experimental results for Solomon's benchmark test problems are presented and analyzed. Moreover, we discuss how heuristic methods should be evaluated and propose using the concept of Pareto optimality in the comparison of different heuristic approaches. The metaheuristic methods are described in the second part of this article.

Key words: vehicle routing; time windows; heuristics; local search metaheuristics; tabu search; genetic algorithms
History: Received: December 2001; revision received: December 2002; accepted: December 2002.

Transportation is an important domain of human activity. It supports and makes possible most other social and economic activities. Whenever we use a telephone, shop at a food store, read our mail, or fly for business or pleasure, we are the beneficiaries of some system that has routed messages, goods, or people from one place to another. Freight transportation, in particular, is one of today's most important activities. Let us mention that the annual cost of excess travel in the United States has been estimated at some $45 billion (King and Mast 1997), and the turnover of goods transportation in Europe is some $168 billion per year. In the United Kingdom, France, and Denmark, for example, transportation represents some 15%, 9%, and 15% of national expenditures, respectively (Crainic and Laporte 1997, Larsen 1999). It is estimated that distribution costs account for almost half of the total logistics costs and in some industries, such as in the food and drink business, distribution costs can account for up to 70% of the value added costs of goods (De Backer et al. 1997, Golden and Wasil 1987). Halse (1992) reports that in 1989, 76.5% of all the transportation of goods was done by vehicles, which also underlines the importance of routing and scheduling problems.

The vehicle routing problem with time windows (VRPTW) is an important problem occurring in many distribution systems. The VRPTW can be defined as follows. Let $G = (V, E)$ be a connected digraph consisting of a set of $n + 1$ nodes, each of which can be serviced only within a specified time interval or time window and a set E of arcs with nonnegative weights, d_{ij}, and with associated travel times, t_{ij}. The travel time, t_{ij}, includes a service time at node i, and a vehicle is permitted to arrive before the opening of the time window, and wait at no cost until service becomes possible, but it is not permitted to arrive after the latest time window. Node 0 represents the depot. Each node i, apart from the depot, imposes a service requirement, q_i, that can be a delivery from or a pickup for the depot. In most of the surveyed papers the objective is to find the minimum number of tours, K^*, for a set of identical vehicles such that each node is reached within its time window and the accumulated service up to any node does not exceed a positive number Q (vehicle capacity). A secondary objective is often either to minimize the total distance traveled or the duration of the routes. All problem parameters, such as customer demands and time windows, are assumed to be

known with certainty. Moreover, each customer must be served by exactly one vehicle, thus prohibiting split service and multiple visits. The tours correspond to feasible routes starting and ending at the depot. Some of the most useful applications of the VRPTW include bank deliveries, postal deliveries, industrial refuse collection, national franchise restaurant services, school bus routing, security patrol services, and JIT (just in time) manufacturing.

The VRPTW has been the subject of intensive research efforts for both heuristic and exact optimization approaches. Early surveys of solution techniques for the VRPTW can be found in Golden and Assad (1986, 1988), Desrochers et al. (1988), and Solomon and Desrosiers (1988). Desrosiers et al. (1995) and Cordeau et al. (2001) focus mainly on exact techniques. Further details on these exact methods can be found in Larsen (1999) and Cook and Rich (1999). Because of the high complexity level of the VRPTW and its wide applicability to real-life situations, solution techniques capable of producing high-quality solutions in limited time, i.e., heuristics, are of prime importance. Over the last few years, many authors have proposed new heuristic approaches, primarily metaheuristics, for tackling the VRPTW. To our knowledge, these have not been comprehensively surveyed and compared. The purpose of this two-part survey is to fill this gap. In the first part, we examine traditional heuristic approaches, that is, route construction and route improvement (local search) methods. These are of interest by themselves because they can provide good solutions with a low computational effort, but also because they are a major component of all metaheuristics for the VRPTW. Metaheuristics are discussed in the second part of this survey.

The remainder of this paper is organized as follows. Section 1 is devoted to a discussion of how heuristics are to be evaluated. Route construction techniques are reviewed in §2 and route improvement (local search) methods in §3. Finally, §4 concludes the paper.

1. Evaluation of Heuristics

Evaluation of any heuristic method is subject to the comparison of a number of criteria that relate to various aspects of algorithm performance. Examples of such criteria are running time, quality of solution, ease of implementation, robustness, and flexibility (Barr et al. 1995, Cordeau et al. 2002). Because heuristic methods are ultimately designed to solve real-world problems, flexibility is an important consideration. An algorithm should be able to easily handle changes in the model, the constraints, and the objective function. As for robustness, an algorithm should not be overly sensitive to differences in problem characteristics: A robust heuristic should not perform poorly on any instance. Moreover, an algorithm should be able to produce good solutions every time it is applied to a given instance. This is to be highlighted because many heuristics are nondeterministic, and contain some random components such as randomly chosen parameter values. The output of separate executions of these nondeterministic methods on the same problem is, in practice, never the same. This makes it difficult to analyze and compare results. Using only the best results of a nondeterministic heuristic, as is often done in the literature, may create a false picture of its real performance. Thus, we consider average results based on multiple executions on each problem an important basis for the comparison of nondeterministic methods. Furthermore, it would also be important to report the worst-case performance. Extensive discussions on these subjects can be found in Cordeau et al. (2002).

The time a heuristic takes to produce good quality solutions can be crucial when choosing between different techniques. Similarly, the quality of the final solution, as measured by the objective function, is important. How close the solution is to the optimal solution is a standard measure of quality, or if the heuristic is designed to simply produce feasible solutions, then the ability of the heuristic to provide such solutions is important.

There is generally a trade-off between run time and solution quality—the longer a heuristic is run, the better the quality of the final solution. A compromise is needed so that good quality solutions are produced in a reasonable amount of time. Basically, this trade-off between run time and solution quality can be viewed in terms of a multiobjective optimization in which the two objectives are balanced. Performance measures for heuristics can be plotted in two-dimensional space, with the first dimension corresponding to run time and the second to solution quality. In that space, points such that there exist no other points with better values on both dimensions are said to be Pareto optimal; they define effective compromises between the objectives. This is illustrated in Figure 10 of §3, where points Antes and Derigs (1995), Russell (1995), and Bräysy (2003) are the Pareto optimal ones. The choice between different heuristic approaches yielding Pareto optimal results depends on the preferences of the decision maker and the situation at hand.

By far, the most common method of evaluating the solution quality of a heuristic algorithm is empirical analysis. In general, empirical analysis involves testing the heuristic across a wide range of problem instances to get an idea of overall performance. To arrive at conclusions that have meaning in a statistical sense, experimental design should ideally be used on

different levels of the various algorithm parameters and the results compared by appropriate techniques.

In the actual comparison of heuristics, one often faces a number of difficulties. The most obvious difficulty is making the competition fair. Differences between machines first come to mind. In this paper, we address this issue by adjusting reported running times by the factors given by Dongarra (1998). Even more difficult issues to address are differences in coding skill, tuning, and effort invested (Hooker 1995).

Another difficulty faced, especially in the VRPTW context, is that often only the best results obtained during the whole computational study are reported. Moreover, in some cases the authors do not report the number of runs or CPU time required to get the reported results. In these cases it is impossible to conclude anything about the efficiency of the methods, or compare these methods with other approaches. The only adequate basis for comparison of these methods would be optimal solutions, because if enough time is available, it is always preferable to solve the problems to optimality using exact methods. To be able to reach proper conclusions, in addition to the number of runs and time consumption, one should answer questions such as what are the limits of the given algorithm, i.e., how good are the best results that can be obtained using the particular approach, and how good a solution can be obtained in a given amount of time. One should, in other words, report results obtained using different computation times, and observe how much time is needed to obtain results of a given quality. Moreover, in our opinion, figures describing the relationship between solution quality and computation time would greatly facilitate the analysis. Taillard (2001) extensively discusses this issue and proposes reporting an absolute computational effort, such as the number of iterations instead of computating times, and using probability diagrams based on repeated Mann-Whitney statistical tests. Obviously, such an approach is not possible when one relies on previously published results as we do here.

In the VRPTW context, the most common way to compare heuristics is the results obtained for Solomon's (1987) 56 benchmark problems. These problems have 100 customers, a central depot, capacity constraints, time windows on the time of delivery, and a total route time constraint. The C1 and C2 classes, have customers located in clusters, and in the R1 and R2 classes the customers are at random positions. The RC1 and RC2 classes contain a mix of both random and clustered customers. Each class contains between 8 and 12 individual problem instances, and all problems in any one class have the same customer locations and the same vehicle capacities; only the time windows differ. In terms of time window density (the percentage of customers with time windows), the problems have 25%, 50%, 75%, and 100% time windows. The C1, R1, and RC1 problems have a short scheduling horizon and require 9 to 19 vehicles. Short horizon problems have vehicles that have small capacities and short route times, and cannot service many customers at one time. Classes C2, R2, and RC2 are more representative of "long-haul" delivery with longer scheduling horizons and fewer (two to four) vehicles. Both travel time and distance are given by the Euclidean distance between points.

The results are usually ranked according to a hierarchical objective function, where the number of vehicles is considered as the primary objective, and for the same number of vehicles, the secondary objective is often either total traveled distance or total duration of routes. Therefore, a solution requiring fewer routes is always considered better than a solution with more routes, regardless of the total traveled distance. According to Bräysy (2001) these two objectives are very often conflicting, meaning that the reduction in number of vehicles often causes increase in total traveled distance. Thus, a better solution in terms of total distance can be obtained by increasing the number of routes. Some other papers report similar findings; see, for example, Caseau and Laburthe (1999).

2. Route Construction Heuristics

Route construction heuristics select nodes (or arcs) sequentially until a feasible solution has been created. Nodes are chosen based on some cost minimization criterion, often subject to the restriction that the selection does not create a violation of vehicle capacity or time window constraints. Sequential methods construct one route at a time, while parallel methods build several routes simultaneously.

Solomon (1986) proposes a so-called route-first cluster-second scheme using a giant-tour heuristic. First, the customers are scheduled into one giant tour, which is then divided into a number of smaller routes. The initial giant tour could, for example, be generated as a traveling salesman tour without considering the capacity and time constraints. No computational results are given in the paper for the heuristic.

Solomon (1987) describes several heuristics for the VRPTW. One of the methods is an extension to the savings heuristic of Clarke and Wright (1964). The savings method, originally developed for the classical VRP, is probably the best-known route construction heuristic. It begins with a solution in which every customer is supplied individually by a separate route. Combining the two routes serving, customers i and j, respectively, results in a cost savings of $S_{ij} = d_{i0} + d_{0j} - d_{ij}$. Clarke and Wright (1964) select the arc (i, j) linking customers i and j with maximum S_{ij},

subject to the requirement that the combined route is feasible. With this convention, the route combination operation is applied iteratively. In combining routes, one can simultaneously form partial routes for all vehicles or sequentially add customers to a given route until the vehicle is fully loaded. To account for both the spatial and temporal closeness of customers, Solomon sets a limit to the waiting time of the route. The savings method is illustrated in Figure 1.

The second heuristic, a time-oriented nearest neighbor, starts every route by finding an unrouted customer closest to the depot. At every subsequent iteration, the heuristic searches for the customer closest to the last customer added into the route and adds it at the end of the route. A new route is started any time the search fails to find a feasible insertion place, unless there are no more unrouted customers left. The metric used to measure the closeness of any pair of customers attempts to account for both geographical and temporal closeness of customers.

The most successful of the three proposed sequential insertion heuristics is called I1. A route is first initialized with a "seed" customer and the remaining unrouted customers are added into this route until it is full, with respect to the scheduling horizon and/or capacity constraint. If unrouted customers remain, the initializations and insertion procedures are then repeated until all customers are serviced. The seed customers are selected by finding either the geographically farthest unrouted customer in relation to the depot or the unrouted customer with the lowest allowed starting time for service. After initializing the current route with a seed customer, the method uses two subsequently defined criteria, $c_1(i, u, j)$ and $c_2(i, u, j)$, to select customer u for insertion between adjacent customers i and j in the current partial route. Let $(i_0, i_1, i_2, \ldots, i_m)$ be the current route, with i_0 and i_m representing the depot. For each unrouted customer u, we first compute its best feasible insertion cost on the route as

$$c_1(i(u), u, j(u)) = \min_{\rho=1,\ldots,m} c_1(i_{\rho-1}, u, i_\rho), \qquad (1)$$

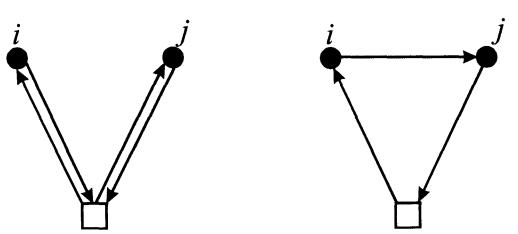

Figure 1 The Savings Heuristic

Note. In the left part, customers i and j are served by separate routes; in the right part, the routes are combined by inserting customer j after i.

Next, the best customer u^* to be inserted in the route is the one for which

$$c_2(i(u^*), u^*, j(u^*)) = \max_u \{c_2(i(u), u, j(u)) \mid u \text{ is unrouted and the route is feasible}\}. \qquad (2)$$

Client u^* is then inserted into the route between $i(u^*)$ and $j(u^*)$. When no more customers with feasible insertions can be found, the method starts a new route, unless it has already routed all customers. More precisely $c_1(i, u, j)$ is calculated as

$$c_1(i, u, j) = \alpha_1 c_{11}(i, u, j) + \alpha_2 c_{12}(i, u, j), \qquad (3)$$

where

$$\alpha_1 + \alpha_2 = 1, \quad \alpha_1 \geq 0, \quad \alpha_2 \geq 0,$$

$$c_{11}(i, u, j) = d_{iu} + d_{uj} - \mu d_{ij}, \quad \mu \geq 0, \qquad (4)$$

$$c_{12}(i, u, j) = b_{ju} - b_j, \qquad (5)$$

and d_{iu}, d_{uj}, and d_{ij} are distances between customers i and u, u and j, and i and j, respectively. Parameter μ controls the savings in distance, and b_{ju} denotes the new time for service to begin at customer j, given that u is inserted on the route and b_j is the beginning of service before insertion. The criterion $c_2(i, u, j)$ is calculated as follows

$$c_2(i, u, j) = \lambda d_{0u} - c_1(i, u, j), \quad \lambda \geq 0. \qquad (6)$$

Parameter λ is used to define how much the best insertion place for an unrouted customer depends on its distance from the depot, and on the other hand how much the best place depends on the extra distance and extra time required to visit the customer by the current vehicle. The second type of proposed insertion heuristics (I2) aims to select customers whose insertion costs minimize a measure of total route distance and time, and the third approach (I3) accounts for the urgency of servicing a customer.

Dullaert (2000) and Dullaert and Bräysy (2003) argue that Solomon's time insertion criterion $c_{12}(i, u, j)$ underestimates the additional time needed to insert a new customer u between the depot and the first customer in the partially constructed route. This can cause the insertion criterion to select suboptimal insertion places for unrouted customers. Thus, a route with a relatively small number of customers can have a larger schedule time than necessary. The author introduces new time insertion criteria to solve the problem and concludes that the new criteria offer significant cost savings starting from more than 50%. These cost savings, however, decrease as the number of customers per route increases.

Solomon's (1987) time-oriented sweep heuristic is based on the idea of decomposing the problem into

a clustering stage and a scheduling stage. In the first phase, customers are assigned to vehicles as in the original sweep heuristic (Gillett and Miller 1974). Here, a "center of gravity" is computed and the customers are partitioned according to their polar angle. In the second phase, customers assigned to a vehicle are scheduled using an insertion heuristic of type I1.

Potvin and Rousseau (1993) introduce a parallel version of Solomon's insertion heuristic I1, where the set of m routes is initialized at once. The authors use Solomon's sequential insertion heuristic to determine the initial number of routes and the set of seed customers. The selection of the next customer to be inserted is based on a generalized regret measure over all routes. A large regret measure means that there is a large gap between the best insertion place for a customer and its best insertion places in the other routes.

Foisy and Potvin (1993) implemented the above-described parallel version of Solomon's insertion heuristic on parallel hardware consisting of two to six Sun 3 workstation transputers. The parallelism is exploited in the calculation of insertion cost for each customer. The selection of the best customer for insertion is then run only on half of the available processors. To reduce the unequal workload among the processors, unrouted customers are reassigned among the processors, so as to reduce the average processor's idle time. The authors conclude that the overall reduction in computation time is linear with the number of processors for the distributed part of the heuristic algorithm.

Ioannou et al. (2001) use the generic sequential insertion framework proposed by Solomon (1987) to solve a number of theoretical benchmark problems and an industrial example from the food industry. The proposed approach is based on new criteria for customer selection and insertion that are motivated by the minimization function of the greedy look-ahead solution approach of Atkinson (1994). The basic idea behind the criteria is that a customer u selected for insertion into a route should minimize the impact of the insertion on the customers already on the route under construction, on all nonrouted customers, and on the time window of customer u, himself.

Balakrishnan (1993) describes three heuristics for the vehicle routing problem with soft time windows (VRPSTW). The heuristics are based on nearest neighbor and Clarke-Wright savings rules, and they differ only in the way used to determine the first customer on a route and in the criteria used to identify the next customer for insertion. The motivation behind VRPSTW is that by allowing limited time window violations for some customers, it may be possible to obtain significant reductions in the number of vehicles required and/or the total distance or time of all routes. Among the soft time window problem instances, dial-a-ride problems play a central role.

Bramel and Simchi-Levi (1996) propose an asymptotically optimal heuristic based on the idea of solving the capacitated location problem with time windows (CLPTW). In CLPTW, the objective is to select a subset of possible sites, to locate one vehicle to each site, and to assign customers to the vehicles. In the VRPTW context, this selection of locations for vehicles refers to selecting a set of seed customers that initialize the routes. The authors use a Lagrangian relaxation-based technique to solve the CLPTW and the other customers are inserted in greedy order into simple tours by favoring customers that least increase the distance traveled. The authors conclude that their heuristic provides a better solution than Solomon's heuristic for 25 of the 56 problems using reasonable running times.

Table 1 compares some of the described route construction algorithms. The first column to the left indicates the authors. Columns R1, R2, C1, C2, RC1, and RC2 present the average number of vehicles and average total distance with respect to the six problem groups of Solomon (1987). Finally, the rightmost column indicates the cumulative number of vehicles and cumulative total distance over all 56 test problems. In the lower part of the table, we report information regarding the computer used, number of runs, and average time consumption in minutes as reported by

Table 1 Route Construction Heuristics

Author	R1	R2	C1	C2	RC1	RC2	CNV/CTD
(1) Solomon (1987)	13.58	3.27	10.00	3.13	13.50	3.88	453
	1,436.7	1,402.4	951.9	692.7	1,596.5	1,682.1	73,004
(2) Potvin et al. (1993)	13.33	3.09	10.67	3.38	13.38	3.63	453
	1,509.04	1,386.67	1,343.69	797.59	1,723.72	1,651.05	78,834
(3) Ioannou et al. (2001)	12.67	3.09	10.00	3.13	12.50	3.50	429
	1,370	1,310	865	662	1,512	1,483	67,891

Note. For all algorithms, the average results for Solomon's benchmarks are described. The notations CNV and CTD in the rightmost column indicate the cumulative number of vehicles and cumulative total distance over all 56 test problems.

(1) DEC 10, 1 run, 0.6 min.; (2) IBM PC, 1 run, 19.6 min.; (3) Intel Pentium 133 MHz, 1 run, 4.0 min.

the authors. We could not include all the algorithms described in the table due to lack of information (not all authors report results properly or use Solomon's problem set). In Table 1, the number of vehicles is the primary minimization objective, and the secondary objective is the total duration of routes in Solomon (1987) and Potvin and Rousseau (1993) and the total distance in Ioannou et al. (2001). Thus, in some cases, there may be a slight overestimation of the total distance values of Solomon (1987) and Potvin and Rousseau (1993). The methods by Solomon (1987) and Potvin and Rousseau (1993) are coded in Fortran; Ioannou et al. (2001) do not report the programming language used. Finally, because we used rounded distance measures reported by other authors to calculate the cumulative total distance (CTD), we rounded the values to integers in Tables 1 and 2.

It seems that Ioannou et al. (2001) produce the best results, though at the cost of higher computation times. As for the other two methods, Solomon (1987) seems to be better than Potvin and Rousseau (1993) in clustered problem groups C1 and C2, while the opposite is true for the other problem groups. These heuristics are very fast, and there are no significant differences in the computational burden if one takes into account the differences in the hardware used. Compared to local search approaches, these construction heuristics are considerably faster, as one can see from Figure 10. However, these simple procedures lack in solution quality compared to more sophisticated approaches.

3. Solution Improvement Methods

Classical local search methods form a general class of approximate heuristics based on the concept of iteratively improving the solution to a problem by exploring neighboring ones. To design a local search algorithm, one typically needs to specify the following choices: how an initial feasible solution is generated, what move-generation mechanism to use, the acceptance criterion, and the stopping test. The move-generation mechanism creates the neighboring solutions by changing one attribute or a combination of attributes of a given solution. Here attribute could refer, for example, to arcs connecting a pair of customers. Once a neighboring solution is identified, it is compared against the current solution. If the neighboring solution is better, it replaces the current solution, and the search continues. Two acceptance strategies are common in the VRPTW context, namely first-accept (FA) and best-accept (BA). The first-accept strategy selects the first neighbor that satisfies the predefined acceptance criterion. The best-accept strategy examines all neighbors satisfying the criterion and selects the best among them.

Table 2 Local Search Algorithms

Author	R1	R2	C1	C2	RC1	RC2	CNV/CTD
(1) Thompson et al. (1993)	13.00	3.18	10.00	3.00	13.00	3.71	438
	1,356.92	1,276.00	916.67	644.63	1,514.29	1,634.43	6,8916
(2) Potvin et al. (1995)	13.33	3.27	10.00	3.13	13.25	3.88	448
	1,381.9	1,293.4	902.9	653.2	1,545.3	1,595.1	69,285
(3) Russell (1995)	12.66	2.91	10.00	3.00	12.38	3.38	424
	1,317	1,167	930	681	1,523	1,398	65,827
(4) Antes et al. (1995)	12.83	3.09	10.00	3.00	12.50	3.38	429
	1,386.46	1,366.48	955.39	717.31	1,545.92	1,598.06	71,158
(5) Prosser et al. (1996)	13.50	4.09	10.00	3.13	13.50	5.13	471
	1,242.40	977.12	843.84	607.58	1,408.76	1,111.37	58,273
(6) Shaw (1997)	12.31	—	10.00	—	12.00	—	—
	1,205.06		8.28.38		1,360.40		
(7) Shaw (1998)	12.33	—	10.00	—	11.95	—	—
	1,201.79		828.38		1,364.17		
(8) Caseau et al. (1999)	12.42	3.09	10.00	3.00	12.00	3.38	420
	1,233.34	990.99	828.38	596.63	1,403.74	1,220.99	58,927
(9) Schrimpf et al. (2000)	12.08	2.82	10.00	3.00	11.88	3.38	412
	1,211.53	949.27	828.38	589.86	1,361.76	1,097.63	56,830
(10) Cordone et al. (2001)	12.50	2.91	10.00	3.00	12.38	3.38	422
	1,241.89	995.39	834.05	591.78	1,408.87	1,139.70	58,481
(11) Bräysy (2003)	12.17	2.82	10.00	3.00	11.88	3.25	412
	1,253.24	1,039.56	832.88	593.49	1,408.44	1,244.96	59,945

Note. For each method two average results for Solomon's benchmarks are presented. The rightmost CNV and CTD indicate the cumulative number of vehicles and cumulative total distance over all test problems.

(1) PC/AT 12 MHz, 4 runs, 1.8 min.; (2) Sparc workstation, –, 3.0 min.; (3) PC/486/DX2 66 MHz, 3 runs, 1.4 min.; (4) RS6000/530, 4 runs, 3.6 min.; (5) –, –, –; (6) DEC Alpha, 3 runs, 2 hours; (7) Sun Ultra Sparc 143 MHz, 6 runs, 1 hour; (8) Pentium 300 MHz, 1 run, 5 min.; (9) RS 6000, –, 30 min.; (10) Pentium 166 MHz, 1 run, 15.7 min.; (11) Pentium 200 MHz, 1 run, 4.6 min.

The local optimum produced by any local search procedure can be very far from the optimal solution. Local search methods perform a myopic search because they only sequentially accept solutions that produce reductions in the objective function value. Thus, the outcome depends heavily on initial solutions and the neighborhood generation mechanism used. Most iterative improvement methods that have been applied to vehicle routing and scheduling problems are edge-exchange algorithms.

The edge-exchange neighborhoods for a single route are the set of tours that can be obtained from an initial tour by replacing a set of k of its edges by another set of k edges. Such replacements are called k-exchanges, and a tour that cannot be improved by a k-exchange is said to be k-optimal. Verifying k-optimality requires $O(n^k)$ time. Figure 2 illustrates 2-exchange or 2-opt. It tries to improve the tour by replacing two of its edges by two other edges and iterates until no further improvement is possible.

Russell (1977) reports early work on the VRPTW for a k-optimal improvement heuristic. The so-called M-tour approach was effective in solving an actual problem with a few time-constrained customers. A solution for a 163-customer problem with 15% time-constrained customers was generated in less than 90 seconds of IBM 370/168 CPU time.

Efficient implementations for speeding up the screening of infeasible solutions and the evaluation of the objective function are reported in Savelsbergh (1986), Solomon and Desrosiers (1988), Solomon et al. (1988), Savelsbergh (1990), and Savelsbergh (1992). The techniques used involve preprocessing, tailored updating mechanisms, and lexicographic search strategies.

Baker and Schaffer (1986) report on a computational study of route improvement procedures, which are applied to heuristically generated initial solutions. Time-oriented nearest neighbor and three different cheapest insertion algorithms with differing cost functions are used for solution construction purposes. The cost functions consider one or more of the following components: distance, increase in arrival time, and waiting time. The improvement methods considered are extensions to the VRPTW of the 2-opt and 3-opt edge exchange procedures of Lin (1965). Both intraroute and interroute exchanges are tested. The authors conclude that the best overall solutions are usually obtained from the best starting solutions, and that, generally, the cheapest insertion procedures outperformed the nearest neighbor ones. The authors also conclude that only less than 10% of the solution improvements involve the reversal of the orientation of a sequence of two or more customers.

Van Landeghem (1988) presents a bicriteria heuristic based on the savings method of Clarke and Wright (1964). More precisely, the author proposes combining the original savings measure in terms of travel time with so-called "loss of flexibility." The flexibility is defined as the difference between customer time window length and route time window length after combining. Route time window refers to the difference between time slots, inside which a vehicle can start servicing the first and last customers on the route. In the end, the results are improved using simple customer reinsertions. A closely related operator is the Or-opt introduced by Or (1976) for the traveling salesman problem. The basic idea is to relocate a chain of l consecutive vertices (customers). This is achieved by replacing three edges in the original tour by three new ones without modifying the orientation of the route as illustrated in Figure 3.

Koskosidis et al. (1992) describe an extension of the cluster-first, route-second algorithm of Fisher and Jaikumar (1981) for the variant of the VRP with soft time window constraints, where the time windows can be violated at a cost. The problem is heuristically decomposed into a capacitated clustering problem and a series of traveling salesman problems with soft time windows. The clustering problem is solved with a greedy heuristic procedure that assigns customers to selected seeds according to a regret function representing the penalty of not assigning the customer to its closest seed. Then, an attempt is made to find new

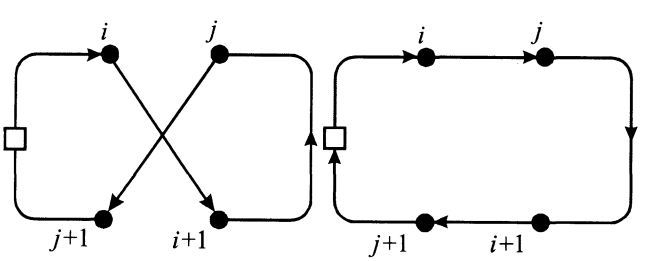

Figure 2 2-Opt Exchange Operator

Note. The edges $(i, i+1)$ and $(j, j+1)$ are replaced by edges (i, j) and $(i+1, j+1)$, thus reversing the direction of customers between $i+1$ and j.

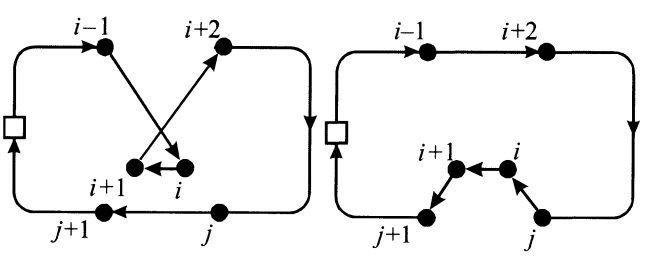

Figure 3 The Or-Opt Operator

Note. Customers i and $i+1$ are relocated to be served between two customers j and $j+1$, instead of customers $i-1$ and $i+2$. This is performed by replacing three edges $(i-1, i)$, $(i+1, i+2)$, and $(j, j+1)$ by the edges $(i-1, i+2)$, (j, i), and $(i+1, j+1)$, preserving the orientation of the route.

seed customers with lower clustering cost, and local exchanges between all customer pairs belonging to different clusters are performed. For the routing part, the authors propose a combination of a total enumeration algorithm, a reduced gradient scheduling algorithm, and the branch exchange heuristic of Lin and Kernighan (1973). The iterative master-slave solution procedure approximates linearly the clustering costs and improves the approximation successively through the actual routing costs obtained. Numerical results based on randomly generated and benchmark problem sets indicate that the algorithm outperforms Solomon's insertion heuristic and 2-opt and 3-opt improvement heuristics of Baker and Schaffer (1986), though at the cost of a clearly higher computational effort.

Potvin and Rousseau (1995) compare different edge exchange heuristics for VRPTW (2-opt, 3-opt, and Or-opt) and introduce a new 2-opt* exchange heuristic. The basic idea in 2-opt* is to combine two routes so that the last customers of a given route are introduced after the first customers of another route, thus preserving the orientation of the routes. The operator is illustrated in Figure 4, where the edges $(i, i+1)$ and $(j, j+1)$ are replaced by $(i, j+1)$ and $(j, i+1)$, i.e., the end portions of two routes are exchanged. As a special case, it can combine two routes into one if edge $(i, i+1)$ is the first one on its route and edge $(j, j+1)$ the last one on its route or vice versa. A hybrid approach based on Or-opt and 2-opt* exchanges is found to be particularly powerful. This approach oscillates between the two neighborhoods by changing the operator each time local minimum is found. The authors also test an implementation where the two operators are merged together. The initial solutions are created with Solomon's I1 heuristic.

Prosser and Shaw (1996) compare intraroute 2-opt by Lin (1965) and interroute operators relocate, exchange, and cross, originally proposed by Savelsbergh (1992) for the classical VRP. The 2-opt works by reversing part of a single route (see Figure 1). The relocate operator simply moves a customer visit from one route to another. It is illustrated in Figure 5.

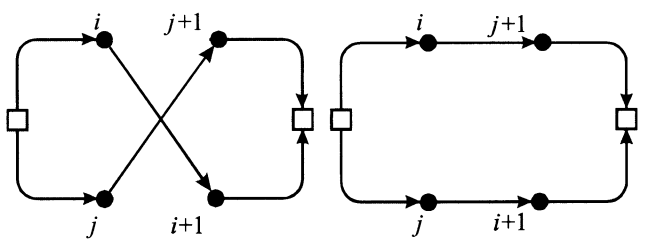

Figure 4 2-Opt* Operator

Note. The customers served after customer i on the upper route are reinserted to be served after customer j on the lower route, and customers after j on the lower route are moved to be served on the upper route after customer i. This is performed by replacing edges $(i, i+1)$ and $(j, j+1)$ with edges $(i, j+1)$ and $(j, i+1)$.

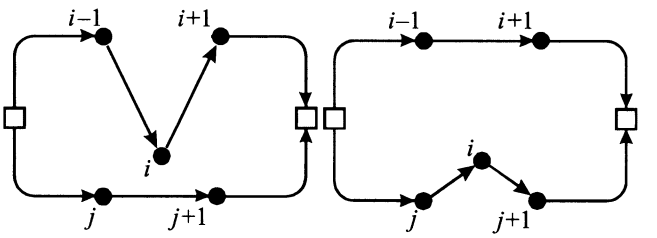

Figure 5 Relocate Operator

Note. The edges $(i-1, i)$, $(i, i+1)$, and $(j, j+1)$ are replaced by $(i-1, i+1)$, (j, i), and $(i, j+1)$, i.e., customer i from the origin route is placed into the destination route.

The exchange heuristic swaps two visits in different routes. This is pictured in Figure 6. Finally, cross is similar to 2-opt* proposed by Potvin and Rousseau (1995) for VRPTW. Initially, a virtual vehicle, which performs the visits not carried out by the real vehicles, exists. This virtual vehicle is different from the real ones in two respects. First, the virtual vehicle can make an unlimited number of customer visits. Second, the cost incurred by the virtual vehicle when it performs a customer visit is typically higher than that incurred by a real vehicle.

De Backer et al. (1997) report research similar to Prosser and Shaw (1996) in the constraint programming (CP) context. CP is a paradigm for representing and solving a wide variety of problems. Tackling combinatorial problems generally involves manipulating variables that can take a finite number of values. In CP, a domain is associated with every variable. The domain is created by using constraints on variables that restrict the possible combinations of values for the variables. Looking locally at a particular constraint, the CP algorithm attempts to remove from the domain of each variable involved in that constraint values that cannot be part of any solution. This reduction of a variable's domain triggers the examination of all constraints involving this variable, which in turn

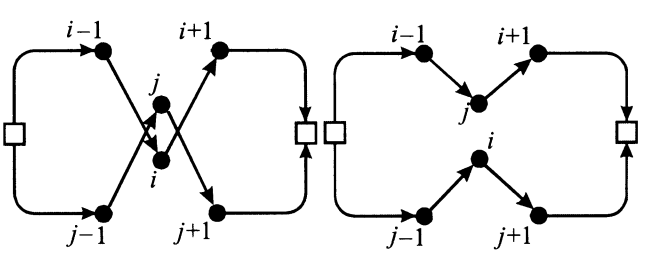

Figure 6 The Exchange Operator

Note. The edges $(i-1, i)$, $(i, i+1)$, $(j-1, j)$, and $(j, j+1)$ are replaced by $(i-1, j)$, $(j, i+1)$, $(j-1, i)$, and $(i, j+1)$, i.e., two customers from different routes are simultaneously placed into the other routes.

may reduce other domains. This recursive process stops when either no new domain reduction has taken place or a domain becomes empty. In constraint programming, the computation is driven by constraints, thus giving them an active role. The problems are then solved using often complete search techniques such as depth-first search and branch and bound. For more details on CP, see, for example, Jaffar and Lassez (1986) and Jaffar and Maher (1994).

Other frequently applied neighborhood operators are the λ-interchange of Osman (1993), the CROSS-exchange of Taillard et al. (1997), the GENI-exchange of Gendreau et al. (1992), and ejection chains (Glover 1991, 1992). The λ-exchange generation mechanism can be described as follows: Given a solution for the problem represented by a set of routes $S = \{r_1, \ldots, r_p, \ldots, r_q, \ldots, r_k\}$, a λ-interchange between a pair of routes (r_p, r_q) is a replacement of a subset of customers $S_1 \subseteq r_p$ of size $|S_1| \leq \lambda$ by another subset $S_2 \subseteq r_q$ of size $|S_2| \leq \lambda$ to get two new routes $r_p = (r_p - S_1) \cup S_2$ and $r_q = (r_q - S_2) \cup S_1$ and a new neighboring solution $S' = \{r_1, \ldots, r_p, \ldots, r_q, \ldots, r_k\}$. The neighborhood $N_\lambda(S)$ of a given solution S is the set of all neighbors S' generated in this way for a given value of λ.

In CROSS-exchanges, the basic idea is to first remove two edges $(i-1, i)$ and $(k, k+1)$ from a first route, while two edges $(j-1, j)$, and $(l, l+1)$ are removed from a second route. Then the segments $i-k$ and $j-l$, which may contain an arbitrary number of customers, are swapped by introducing the new edges $(i-1, j)$, $(l, k+1)$, $(j-1, i)$, and $(k, l+1)$ as illustrated in Figure 7.

Ejection chains (Glover 1991, 1992) are based on the notion of generating compound sequences of moves, leading from one solution to another, by linked steps in which changes in selected elements cause other elements to be ejected from their current state, position, or value assignment. In the VRP context, moves refer to the removal of a customer from its route and its reinsertion in another route. The goal

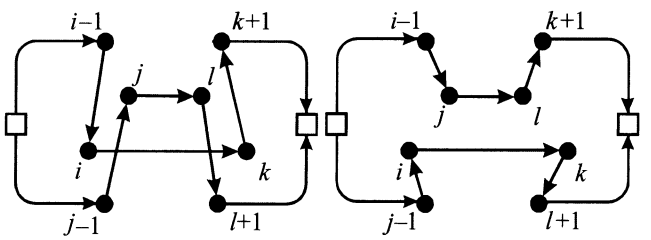

Figure 7 CROSS-Exchange

Note. Segments (i, k) on the upper route and (j, l) on the lower route are simultaneously reinserted into the lower and upper routes, respectively. This is performed by replacing edges $(i-1, i)$, $(k, k+1)$, $(j-1, j)$, and $(l, l+1)$ by edges $(i-1, j)$, $(l, k+1)$, $(j-1, i)$, and $(k, l+1)$. Note that the orientation of both routes is preserved.

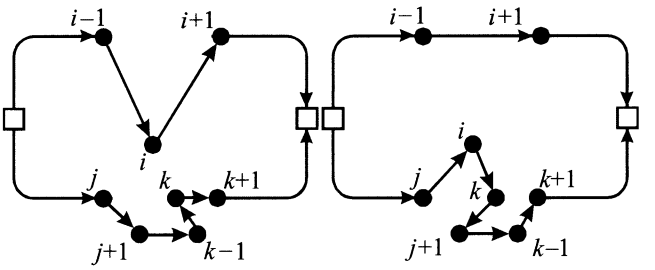

Figure 8 The GENI-Exchange Operator

Note. Customer i on the upper route is inserted into the lower route between the customers j and k closest to it by adding the edges (j, i) and (i, k). Because j and k are not consecutive, one has to reorder the lower route. Here, the feasible tour is obtained by deleting edges $(j, j+1)$ and $(k-1, k)$ and by relocating the path $\{j+1, \ldots, k-1\}$.

is to "make room" for a new customer in a route by first removing another customer from the same route. In each phase within the ejection chain, one customer remains unrouted. The removal and insertion procedures are repeated until one can insert a customer into another route without the need to remove (eject) any customer. The GENI operator is an extension of the relocate neighborhood in which a customer can also be inserted between the two customer nodes on the destination route that are nearest to it, even if these customer nodes are not consecutive. The operator is illustrated in Figure 8.

Thompson and Psaraftis (1993) propose a method based on the concept of cyclic k-transfers that involves simultaneously transferring k demands from route I^j to route $I^{\delta(j)}$ for each j and fixed integer k. The set of routes $\{I^r\}$, $r = 1, \ldots, m$ constitutes a feasible solution, and δ is a cyclic permutation of a subset of $\{1, \ldots, m\}$. In particular, when δ has fixed cardinality C, we obtain a C-cyclic k-transfer. By allowing k dummy demands on each route, demand transfers can be performed among permutations rather than cyclic permutations of routes. Due to the complexity of the cyclic transfer neighborhood search, it is performed heuristically. A general methodology developed by Thompson and Orlin (1989) is used for searching cyclic transfer neighborhoods. They transform the search for negative cost cyclic transfers into a search for negative cost cycles in an auxiliary graph. Savelsbergh's (1986) 2-opt procedure is used to maintain local optimality of the routes at all times, and the initial solutions are constructed using the I1 heuristic of Solomon. The three-cyclic, two-transfer operator is illustrated in Figure 9.

Antes and Derigs (1995) propose a parallel construction approach that constructs and improves several tours, simultaneously. The approach is based on the concept of negotiation between customers and tours. First, each unrouted customer requests a service cost from every tour and sends a proposal to the tour

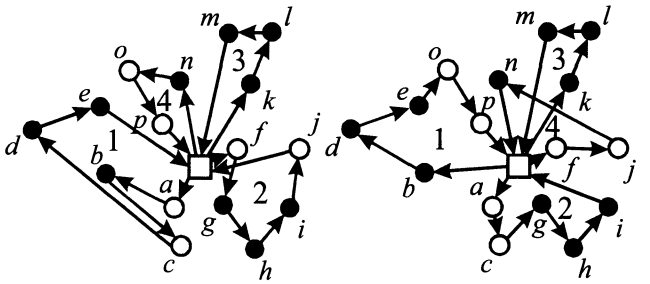

Figure 9 The Cyclic Transfer Operator

Note. The basic idea is to simultaneously transfer the customers denoted by white circles in a cyclical manner between the routes. More precisely, here customers *a* and *c* in Route 1, *f* and *j* in Route 2, and *o* and *p* in Route 4 are simultaneously transferred to Routes 2, 4, and 1, respectively, and Route 3 remains untouched.

that offered the lowest price, and second, each tour selects the most efficient proposal. The prices are calculated according to Solomon's evaluation measures for insertion (heuristic I1). Once a feasible solution is constructed, the number of tours is reduced by one and the problem is resolved. The authors propose also a post-optimization approach, where some of the most inefficient customers are first removed from the tours and then reinserted using the negotiation procedure described above.

Russell (1995) embeds global tour improvement procedures within the tour construction process. The construction procedure used is similar to that in Potvin and Rousseau (1993). N seed points representing fictious customers are first selected using the seed point generation procedure of Fisher and Jaikumar (1981), originally proposed for the classical VRP. The basic idea is to use vehicle capacity information to create sectors and decide the distance of the seeds from the depot within each sector. Three ordering rules are used to select next customer for insertion, namely earliest time window, farthest distance from depot, and width of the time window augmented by distance from the depot. The local search method employs a scheme developed by Christofides and Beasley (1984). In this scheme, a move is performed by deleting and reinserting four customer points close to each other. For each customer, the best two routes are first determined according to the insertion cost of Solomon (1987), because it would be computationally intractable to evaluate all route assignments. This interchange procedure is applied after every k customer has been routed. This approach is compared to the k-opt multiple tour branch exchange heuristic of Russell (1977). The author concludes that the hybrid approach of embedding improvement into the construction procedure is superior compared to the traditional two-phase approach, i.e., route construction followed by solution improvement.

Thangiah et al. (1995) examine the vehicle routing problems with time deadlines (VRPTD), i.e., without earliest time window. They create two heuristics based on principles of time-oriented sweep and cheapest insertion procedures for solving the VRPTD, followed by λ-interchanges of Osman (1993). The authors conclude that the two proposed heuristics perform well for problems in which the customers are tightly clustered or have long deadlines.

Hamacher and Moll (1996) describe a heuristic for real-life VRP's with narrow time windows in the context of delivery of groceries to restaurants. The algorithm is divided into two parts. In the clustering step, the customers are partitioned into regionally bounded sets using the structure of the minimal spanning tree (MST). The MST is divided into subtrees, where nodes of each subtree represent the customers belonging to one tour. Several weight functions based on the number of customers, distance, total demand, and time window types are used to determine whether a subtree leads to a cluster. Then, customers within these sets are routed using a simple cheapest insertion algorithm followed by a local improvement phase, which cuts out pieces of the tour and inserts them back at another feasible location within the same tour. If a feasible solution is not found, the remaining unrouted customers are scheduled manually.

Shaw (1997) describes a large neighborhood search (LNS) based on rescheduling selected customer visits using CP techniques. LNS is analogous to the shuffling technique used in job-shop scheduling (see, for example, Applegate and Cook 1991), which is, itself, inspired from the shifting bottleneck procedure of Adams et al. (1988). The search operates by choosing in a randomized fashion a set of customer visits. The selected customers are removed from the schedule, and then reinserted at optimal cost. To create opportunity for interchange of customer visits between routes, the removed visits are chosen so that they are related. Here, the term related refers to customers that are geographically close to each other, served by the same vehicle, have similar demand for goods and similar starting times for service. A branch-and-bound method coupled with CP is then used to reschedule removed visits. In the initial solution, each customer is served by a separate vehicle. Due to high computational requirements, this approach can be applied only to problems where the number of customers per route is relatively low.

Shaw (1998) uses an LNS approach similar to Shaw's (1997) above approach for solving vehicle routing problems. The basic difference is the usage of constraint-based limited discrepancy search (LDS) in the reinsertion of customers within the branch-and-bound procedure. For more details about LDS, see Harwey and Ginsberg (1995). The number of visits to

be removed is increased during the search each time a number of consecutive attempted moves have not resulted in an improvement of the cost. LDS is used to explore the search tree in order of an increasing number of discrepancies, a discrepancy being a branch against the best insertion places. For instance, a single discrepancy would consist in inserting a customer at its second cheapest position.

Schrimpf et al. (2000) introduce also a methodology similar to LNS that the authors name "ruin and recreate." Three strategies are first used to remove a set of customers from a solution. The removed customers are then reinserted in random order using a greedy cheapest insertion heuristic. The removal procedure removes customers randomly, based on the distance to a randomly selected key customer and a set of succeeding customers on the same route with the key customer. To minimize the number of routes, a fixed penalty is set for routes exceeding the desired minimum number. During the search, solutions that worsen the objective function value are also accepted if the worsening is within a threshold.

Caseau and Laburthe (1999) describe a heuristic specifically designed for large routing problems. The authors introduce an LDS variation to the parallel cheapest insertion heuristic that branches between the best and second best alternative routes for each customer if the differences in insertion costs are small. During solution construction, three moves are considered after each insertion, namely 2-opt*, reinsertion of a chain of consecutive customers from a route r into another route r', as well as a simple customer transfer move. When no feasible insertion place can be found, three different types of moves are considered to make room for the unrouted customer. The first move, swap, removes a chain of consecutive customers from r and inserts it into another route r'. The second move, relocate, removes a vertex from r, and inserts it into another route r', which may recursively require that another vertex is removed from r', etc., followed by reoptimization of each route concerned by the move. The last move, flush and relocate, first removes from r all nodes that can be directly relocated into another route, before trying to insert customer c_i. In cases where the number of customers on a route is less than 30, the order of the customers within the route is optimized using the exact constraint-based branch-and-bound algorithm by Caseau and Laburthe (1997). Otherwise, in case of longer routes, 3-opt is used to modify routes after each insertion. The authors also try to restrict the customers included in each route to a particular geometric zone.

Hong and Park (1999) propose a two-phase heuristic algorithm that consists of a parallel insertion method for clustering and a sequential linear goal programming procedure for routing. The primary criterion for the algorithm is the minimization of the total traveled distance instead of the number of vehicles, and the second criterion is the minimization of the total customer waiting time. The seed customers are selected by identifying customers that cannot be served on the same route due to time or vehicle constraints. The remaining customers are inserted into these initialized tours so that the increase in route distance and waiting time is minimal. Similar to Potvin and Rousseau (1993), customers with a small number of feasible insertion locations and a large difference between the best and next best insertion places are considered for clustering first (regret measure). At the end of the clustering stage, groups are reformed using Or-opt and 2-opt improvement procedures. In the routing stage, the goal programming model is decomposed into two linear programming subproblems, where either total distance or waiting time is minimized first. The authors report slightly better results than Potvin and Rousseau (1993), though using longer computation time.

Cordone and Wolfer-Calvo (2001) propose a deterministic heuristic based on classical k-opt exchanges combined with a procedure to reduce the number of routes. The special feature of the algorithm is that it alternates between minimization of total distance and of total route duration to escape from local minima. The algorithm builds a set of initial solutions using Solomon's insertion heuristic I1, applies a local search procedure (exchanges two or three arcs) to each of them, and chooses the best one. The route reduction procedure tries to insert each customer of one route at a time into another route. If simple insertion fails, a simple ejection chain (Glover 1991, 1992) is tried, where a customer, c_j, is first removed from the target route, r_n, and inserted into some other route, r_m, before inserting the current customer c_i into r_n. The authors use special implementation techniques to reduce the computation time. The first technique is based on so-called macronodes. The macronode is a collapse of whatever sequence of nodes into a single one that is easier to handle (see Cordone and Wolfler-Calvo 1997). The other techniques are exploring the k-neighborhood in lexicographic order (for details, see Savelsbergh 1986) and keeping in mind the best exchange for each route, each pair, and each triplet of routes.

Bräysy (2003) describes several local search heuristics using a new three-phase approach for the VRPTW. In the first phase, several initial solutions are created using the route construction heuristics with different combinations of parameter values. In the second phase, an effort is put to reduce the number of routes using a new ejection chain-based approach (Glover 1991, 1992) that also considers reordering of

the routes. In the third phase, Or-opt exchanges are used to minimize total traveled distance. The first construction heuristics borrow their basic ideas from the studies of Solomon (1987) and Russell (1995). Routes are built, one at a time, in sequential fashion, and after k customers have been inserted into the route, the route is reordered using Or-opt exchanges. In addition, new seed selection schemes are introduced. The second heuristic draws its basic concepts from the savings heuristic of Clarke and Wright (1964). Here, a parallel version of the savings heuristic is implemented, and the original measure of savings is modified to also consider changes in waiting times. Moreover, the customers in the combined route are reordered before evaluating the savings incurred by uniting the two routes.

Table 2 summarizes some of the results obtained by described local search algorithms. We could not include all the algorithms described in the table due to lack of information (not all authors report results properly or use Solomon's problem set). In Table 2, most of the algorithms are deterministic in nature. The only stochastic approaches are that of Russell (1995), Shaw (1997, 1998), and Schrimpf et al. (2000). Russell (1995) and Cordone and Wolfler-Calvo (2001) implemented their algorithm in Fortran, and Potvin and Rousseau (1995), Antes and Derigs (1995), Shaw (1998), and Caseau and Laburthe (1999) used C. Thompson and Psaraftis (1993), Prosser and Shaw (1996), Shaw (1997), and Schrimpf et al. (2000) do not report the software used. The number of vehicles is considered as a primary optimization criterion by all authors except Prosser and Shaw (1996), where only the total distance of the routes is minimized. The secondary objective is total distance in most papers. Thompson and Psaraftis (1993), Potvin and Rousseau (1995), and Russell (1995) optimize the total duration of routes; this may cause an overestimation of the total distance values and should be taken into account in the comparison.

According to Table 2, the methods of Schrimpf et al. (2000) and Bräysy (2003) are the best ones with respect to solution quality. The difference in the cumulative number of vehicles is about 14%, compared to the worst method by Prosser and Shaw (1996). The reason for this can be found in the optimization criteria used: In Prosser and Shaw (1996), only the total distance of the routes is considered. Schrimpf et al. (2000) dominates all other methods for four problem groups. For the easy clustered problem group C1, Shaw (1997, 1998), and Caseau and Laburthe (1999) yield equally good output, and in RC2 Bräysy (2003) performs best. It is difficult to conclude anything regarding the computational effort, as many of the authors do not report the CPU time or the number of runs required to get the reported results. Given the information available, the methods of Russell (1995), Caseau and Laburthe (1999), and Bräysy (2003) appear to be the most efficient ones. It should also be noted that, due to poor performance, Shaw (1997, 1998) do not report the results for the problem groups R2, C2, and RC2. Thus, these two procedures are not comparable with other approaches in terms of robustness.

The efficiency of the described methods is illustrated in Figure 10. In Figure 10 we included only approaches where a sufficient amount of information is provided by the authors. At least the computer, number of computational runs, the time consumption, and the number of vehicles must be reported. From Figure 10, one can see that difference in time consumption between Solomon (1987), Potvin and

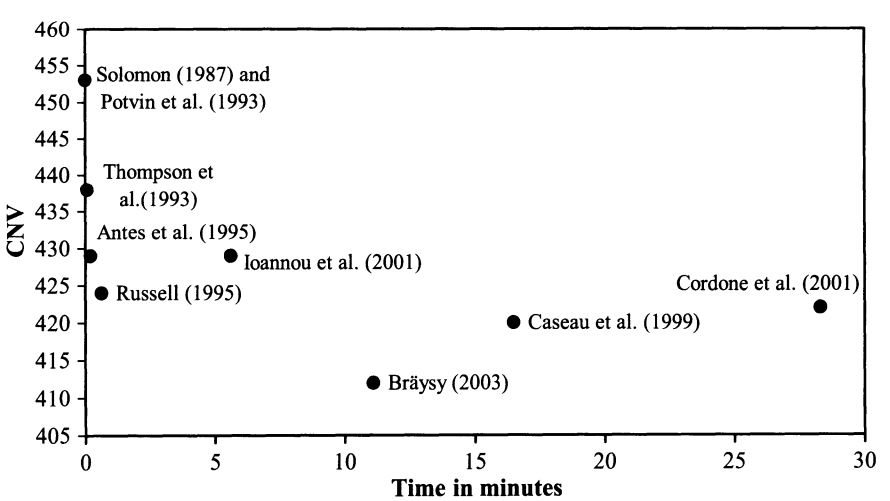

Figure 10 The Efficiency of the Described Methods

Note. The notation CNV refers to the cumulative number of vehicles required to solve all 56 test problems. Note that the time consumption of each method is normalized to Sun Sparc 10 using Dongarra's (1998) factors to facilitate the analysis.

Rousseau (1993), Thompson and Psaraftis (1993), and Antes and Derigs (1995) is quite small. Therefore, only Antes and Derigs (1995), Russell (1995), and Bräysy (2003) can be considered as Pareto optimal in terms of solution quality and time consumption. There is no clear rule to determine which Pareto optimal approach is the best. The choice depends on the preferences of the decision maker. The methods by Antes and Derigs (1995) and Russell (1995) are a lot faster than the one in Bräysy (2003), but they fall behind in solution quality.

4. Conclusions

The vehicle routing problem with time windows is one of the classical research areas in operations research with considerable economic significance. The NP-hardness of the VRPTW requires heuristic solution strategies for most real-life instances. The research on approximation methods has, over the years, produced a wide variety of heuristic approaches for the VRPTW. In this paper, methods based on classical solution construction and improvement techniques were comprehensively reviewed.

VRPTW heuristics are usually measured against two criteria: solution quality in terms of objective function value, and speed. In our opinion, simplicity of implementation, flexibility, and robustness are also essential attributes of good heuristics. By flexibility, we mean the ability to accommodate the various side constraints encountered in a majority of real-life applications. As for robustness, an algorithm should still able to produce results under difficult circumstances, such as when a problem instance is pathological. These issues, as well as the question of how to evaluate heuristics, are discussed in §1.

Recent composite heuristics were found to perform best in terms of solution quality, the most efficient being those of Russell (1995) and Bräysy (2003). These methods provide better results than earlier simple heuristics, while still being quite fast. As heuristics need to be especially effective for very large-scale problems, we expect work on these to intensify.

Acknowledgments

This work was partially supported by the Emil Aaltonen Foundation, the Canadian Natural Science and Engineering Research Council, Liikesivistysrahasto and Volvo Foundations, and TOP project funded by the Research Council of Norway. This support is gratefully acknowledged. The authors also thank Dr. Geir Hasle (SINTEF Applied Mathematics, Norway) for his valuable comments.

References

Adams, J., E. Balas, D. Zawack. 1988. The shifting bottleneck procedure for job shop scheduling. *Management Sci.* **34** 391–401.

Antes, J., U. Derigs. 1995. A new parallel tour construction algorithm for the vehicle routing problem with time windows. Working paper, Department of Economics and Computer Science, University of Köln, Germany.

Applegate, D., W. Cook. 1991. A computational study of the job-shop scheduling problem. *ORSA J. Comput.* **3** 149–156.

Atkinson, J. B. 1994. A greedy look-ahead heuristic for combinatorial optimisation: An application to vehicle scheduling with time windows. *J. Oper. Res. Soc.* **45** 673–684.

Baker, E. K., J. R. Schaffer. 1986. Solution improvement heuristics for the vehicle routing and scheduling problem with time window constraints. *Amer. J. Math. Management Sci.* **6** 261–300.

Balakrishnan, N. 1993. Simple heuristics for the vehicle routeing problem with soft time windows. *J. Oper. Res. Soc.* **44** 279–287.

Barr, R. S., B. L. Golden, J. P. Kelly, M. G. C. Resende, W. R. Stewart. 1995. Designing and reporting on computational experiments with heuristic methods. *J. Heuristics* **1** 9–32.

Bramel, J., D. Simchi-Levi. 1996. Probabilistic analyses and practical algorithms for the vehicle routing problem with time windows. *Oper. Res.* **44** 501–509.

Bräysy, O. 2001. Local search and variable neighborhood search algorithms for the vehicle routing problem with time windows. Doctoral dissertation, University of Vaasa, Finland.

Bräysy, O. 2003. Fast local searches for the vehicle routing problem with time windows. *Inform. Systems Oper. Res.* **41** 179–194.

Caseau, Y., F. Laburthe. 1997. Solving small TSPs with constraints. L. Naish, ed. *Proc. 14th Internat. Conf. Logic Programming*. MIT Press, Cambridge, MA, 316–330.

Caseau, Y., F. Laburthe. 1999. Heuristics for large constrained vehicle routing problems. *J. Heuristics* **5** 281–303.

Christofides, N., J. Beasley. 1984. The period routing problem. *Networks* **14** 237–246.

Clarke, G., J. W. Wright. 1964. Scheduling of vehicles from a central depot to a number of delivery points. *Oper. Res.* **12** 568–581.

Cook, W., J. L. Rich. 1999. A parallel cutting-plane algorithm for the vehicle routing problems with time windows. Technical report TR99-04, Department of Computational and Applied Mathematics, Rice University, Houston, TX.

Cordeau, J. F., G. Desaulniers, J. Desrosiers, M. M. Solomon, F. Soumis. 2001. The VRP with time windows. P. Toth, D. Vigo, eds. *The Vehicle Routing Problem, SIAM Monographs on Discrete Mathematics and Applications*. SIAM, Philadelphia, PA, 157–194.

Cordeau, J.-F., M. Gendreau, G. Laporte, J.-Y. Potvin, F. Semet. 2002. A guide to vehicle routing heuristics. *J. Oper. Res. Soc.* **53** 512–522.

Cordone, R., R. Wolfler-Calvo. 1997. A note on time windows constraints in routing problems. Internal report, Department of Electronics and Information, Polytechnic of Milan, Milan, Italy.

Cordone, R., R. Wolfler-Calvo. 2001. A heuristic for the vehicle routing problem with time windows. *J. Heuristics* **7** 107–129.

Crainic, T. G., G. Laporte. 1997. Planning models for freight transportation. *Eur. J. Oper. Res.* **97** 409–438.

De Backer B., V. Furnon, P. Prosser, P. Kilby, P. Shaw. 1997. Local search in constraint programming: Application to the vehicle routing problem. *Proc. CP-97 Workshop Indust. Constraint-Directed Scheduling*, Schloss Hagenberg, Austria, 1–15.

Desrochers, M., J. K. Lenstra, M. W. P. Savelsbergh, F. Soumis. 1988. Vehicle routing with time windows: Optimization and approximation. B. Golden, A. Assad, eds. *Vehicle Routing: Methods and Studies*. Elsevier Science Publishers, Amsterdam, The Netherlands, 65–84.

Desrosiers, J., Y. Dumas, M. M. Solomon, F. Soumis. 1995. Time constrained routing and scheduling. M. O. Ball, T. L. Magnanti, C. L. Monma, G. L. Nemhauser, eds. *Handbooks in Operations Research and Management Science 8: Network Routing*. Elsevier Science Publishers, Amsterdam, The Netherlands, 35–139.

Dongarra, J. 1998. Performance of various computers using standard linear equations software. Report CS-89-85, Department of Computer Science, University of Tennessee, Knoxville, TN.

Dullaert, W. 2000. Impact of relative route length on the choice of time insertion criteria for insertion heuristics for the vehicle routing problem with time windows. B. Maurizio, ed. *Proc. Rome Jubilee 2000 Conf. Improving Knowledge Tools Transportation Logist.*, Faculty of Engineering, University of Rome, Italy, 153–156.

Dullaert, W. and O. Bräysy. 2003. Routing with relatively few customers per route. *TOP* **11** 325–336.

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

Foisy, C., J.-Y. Potvin. 1993. Implementing an insertion heuristic for vehicle routing on parallel hardware. *Comput. Oper. Res.* **20** 737–745.

Gendreau, M., A. Hertz, G. Laporte. 1992. A new insertion and postoptimization procedures for the traveling salesman problem. *Oper. Res.* **40** 1086–1093.

Gillett, B., L. R. Miller. 1974. A heuristic algorithm for the vehicle dispatch problem. *Oper. Res.* **22** 340–349.

Glover, F. 1991. Multilevel tabu search and embedded search neighborhoods for the traveling salesman problem. Working paper, College of Business and Administration, University of Colorado, Boulder, CO.

Glover, F. 1992. New ejection chain and alternating path methods for traveling salesman problems. O. Balci, R. Sharda, S. Zenios, eds. *Computer Science and Operations Research: New Developments in Their Interfaces*. Pergamon Press, Oxford, U.K., 449–509.

Golden, B. L., A. A. Assad. 1986. Perspectives on vehicle routing: Exciting new developments. *Oper. Res.* **34** 803–809.

Golden, B. L., A. A. Assad, eds. 1988. *Vehicle Routing: Methods and Studies*. Elsevier Science Publishers, Amsterdam, The Netherlands.

Golden, B. L., E. A. Wasil. 1987. Computerized vehicle routing in the soft drink industry. *Oper. Res.* **35** 6–17.

Halse, K. 1992. Modeling and solving complex vehicle routing problems. Ph.D. thesis, Institute of Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark.

Hamacher, A., C. Moll. 1996. A new heuristic for vehicle routing with narrow time windows. U. Derigs, W. Gaul, R. H. Möhring, K.-P. Schuster, eds. *Oper. Res. Proc. 1996, Selected Papers Sympos. (SOR '96)*, Braunschweig, Germany (September 3–6). Springer Verlag, New York, 301–306.

Harwey, W., M. Ginsberg. 1995. Limited discrepancy search. T. Dean, ed. *Proc. 14th IJCAI*. Morgan Kaufmann, San Francisco, CA, 607–615.

Hong, S.-C., Y.-B. Park. 1999. A heuristic for bi-objective vehicle routing with time window constraints. *Internat. J. Production Econom.* **62** 249–258.

Hooker, J. N. 1995. Testing heuristics: We have it all wrong. *J. Heuristics* **1** 33–42.

Ioannou, G., M. Kritikos, G. Prastacos. 2001. A greedy look-ahead heuristic for the vehicle routing problem with time windows. *J. Oper. Res. Soc.* **52** 523–537.

Jaffar, J., J.-L. Lassez. 1986. Constraint logic programming. Technical report 86/73, Department of Computer Science, Monash University, Melbourne, Australia.

Jaffar, J., M. J. Maher. 1994. Constraint logic programming: A survey. *J. Logic Programming* **19/20** 503–581.

King, G. F., C. F. Mast. 1997. Excess travel: Causes, extent and consequences. *Transportation Res. Record* **1111** 126–134.

Kohl, N. 1995. Exact methods for time constrained routing and related scheduling problems. Ph.D. thesis, Institute of Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark.

Kohl, N., J. Desrosiers, O. B. G. Madsen, M. M. Solomon, F. Soumis. 1999. 2-path cuts for the vehicle routing problem with time windows. *Transportation Sci.* **33** 101–116.

Koskosidis, Y. A., W. B. Powell, M. M. Solomon. 1992. An optimization-based heuristic for vehicle routing and scheduling with soft time window constraints. *Transportation Sci.* **26** 69–85.

Larsen, J. 1999. Parallelization of the vehicle routing problem with time windows. Ph.D. thesis, Institute of Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark.

Lin, S. 1965. Computer solutions of the traveling salesman problem. *Bell System Tech. J.* **44** 2245–2269.

Lin, S., B. Kernighan. 1973. An effective heuristic algorithm for the traveling salesman problem. *Oper. Res.* **21** 498–516.

Or, I. 1976. Traveling salesman-type combinatorial problems and their relation to the logistics of regional blood banking. Ph.D. thesis, Northwestern University, Evanston, IL.

Osman, I. H. 1993. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problems. *Ann. Oper. Res.* **41** 421–452.

Potvin, J.-Y., J.-M. Rousseau. 1993. A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. *Eur. J. Oper. Res.* **66** 331–340.

Potvin, J.-Y., J.-M. Rousseau. 1995. An exchange heuristic for routeing problems with time windows. *J. Oper. Res. Soc.* **46** 1433–1446.

Prosser, P., P. Shaw. 1996. Study of greedy search with multiple improvement heuristics for vehicle routing problems. Working paper, University of Strathclyde, Glasgow, Scotland.

Russell, R. 1977. An effective heuristic for the M-tour traveling salesman problem with some side conditions. *Oper. Res.* **25** 517–524.

Russell, R. A. 1995. Hybrid heuristics for the vehicle routing problem with time windows. *Transportation Sci.* **29** 156–166.

Savelsbergh, M. W. P. 1986. Local search in routing problems with time windows. *Ann. Oper. Res.* **4** 285–305.

Savelsbergh, M. W. P. 1990. An efficient implementation of local search algorithms for constrained routing problems. *Eur. J. Oper. Res.* **47** 75–85.

Savelsbergh, M. W. P. 1992. The vehicle routing problem with time windows: Minimizing route duration. *J. Comput.* **4** 146–154.

Schrimpf, G., J. Schneider, H. Stamm-Wilbrandt, G. Dueck. 2000. Record breaking optimization results using the ruin and recreate principle. *J. Comput. Phys.* **159** 139–171.

Shaw, P. 1997. A new local search algorithm providing high quality solutions to vehicle routing problems. Working paper, Department of Computer Science, University of Strathclyde, Glasgow, Scotland.

Shaw, P. 1998. Using constraint programming and local search methods to solve vehicle routing problems. M. Maher, J.-F. Puget, eds. *Principles and Practice of Constraint Programming—CP98, Lecture Notes in Computer Science*. Springer-Verlag, New York, 417–431.

Solomon, M. M. 1986. On the worst-case performance of some heuristics for the vehicle routing and scheduling problem with time window constraints. *Networks* **16** 161–174.

Solomon, M. M. 1987. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Oper. Res.* **35** 254–265.

Solomon, M. M., J. Desrosiers. 1988. Time window constrained routing and scheduling problems. *Transportation Sci.* **22** 1–13.

Solomon, M. M., E. K. Baker, J. R. Schaffer. 1988. Vehicle routing and scheduling problems with time window constraints: Efficient implementations of solution improvement procedures. B. Golden, A. Assad, eds. *Vehicle Routing: Methods and Studies*. Elsevier Science Publishers, Amsterdam, The Netherlands, 85–106.

Taillard, É. D. 2001. Comparison of non-deterministic iterative methods. *MIC '2001, 4th Metaheuristics Internat. Conf.*, Porto, Portugal, (July 16–20).

Taillard, É., P. Badeau, M. Gendreau, F. Guertin, J.-Y. Potvin. 1997. A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation Sci.* **31** 170–186.

Thangiah, S. R., I. H. Osman, R. Vinayagamoorthy, T. Sun. 1995. Algorithms for the vehicle routing problems with time deadlines. *Amer. J. Math. Management Sci.* **13** 323–355.

Thompson, P. M., J. B. Orlin. 1989. Theory of cyclic transfers, Working paper, Operations Research Center, MIT, Cambridge, MA.

Thompson, P. M., H. N. Psaraftis. 1993. Cyclic transfer algorithms for multivehicle routing and scheduling problems. *Oper. Res.* **41** 935–946.

Van Landeghem, H. R. G. 1988. A bi-criteria heuristic for the vehicle routing problem with time windows. *Eur. J. Oper. Res.* **36** 217–226.