A Reactive Variable Neighborhood Search for the Vehicle Routing Problem with Time Windows

Olli Bräysy

SINTEF Applied Mathematics, Department of Optimization, P.O. Box 124 Blindern, N-0314 Oslo, Norway

Olli.Braysy@math.sintef.no

The purpose of this paper is to present a new deterministic metaheuristic based on a modification of Variable Neighborhood Search of Mladenovic and Hansen (1997) for solving the vehicle routing problem with time windows. Results are reported for the standard 100, 200 and 400 customer data sets by Solomon (1987) and Gehring and Homberger (1999) and two real-life problems by Russell (1995). The findings indicate that the proposed procedure outperforms other recent local searches and metaheuristics. In addition four new best-known solutions were obtained. The proposed procedure is based on a new four-phase approach. In this approach an initial solution is first created using new route construction heuristics followed by route elimination procedure to improve the solutions regarding the number of vehicles. In the third phase the solutions are improved in terms of total traveled distance using four new local search procedures proposed in this paper. Finally in phase four the best solution obtained is improved by modifying the objective function to escape from a local minimum.

(*Metaheuristics; Vehicle Routing; Time Windows*)

1. Introduction

The effective management of the distribution of goods and services involves all three levels of strategic, tactical, and operational planning. Decisions relating to the location of facilities, e.g. plants and depots are viewed as strategic, while vehicle fleet size and mix determinations are key tactical issues. Finally, routing and scheduling of vehicles are primarily operational issues in the management of any logistics study. An important problem occurring in many distribution systems is the Vehicle Routing Problem with Time Windows (VRPTW).

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 reached only within a specified time interval or time window, and a set E of arcs with non negative weights and with associated travel times. Let one of the nodes be designated as the depot. Each node i, apart from the depot, imposes a service requirement q_i which can be a delivery from, or a pickup for the depot. The problem is to find the minimum number of tours, K^*, 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 to minimize the total distance traveled. All problem parameters, such as customer demand 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.

The VRPTW is a basic distribution management problem that can be used to model many real-world problems. 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 vendor deliveries for just-in-time manufacturing.

The VRPTW has been the subject of intensive research efforts for both heuristic and optimization approaches. Because of the high complexity of the VRPTW, in practice one must often concentrate on techniques that will produce high quality solutions in limited time. Special purpose surveys on the VRPTW can be found in Golden and Assad (1986), Desrochers et al. (1988), Golden and Assad (1988), Solomon and Desrosiers (1988), Desrosiers et al. (1995) and Cordeau et al. (2001).

Here, we focus on heuristic and metaheuristic approaches. Details about optimization methods can be found for example in the last two of the survey papers mentioned above and the excellent Ph.D. theses by Kohl (1995) and Larsen (1999). For the most successful implementations, see for example Kohl et al. (1999) and Cook and Rich (1999).

A number of route construction methods have been proposed by Solomon (1987). These construction methods build a feasible solution by inserting at every iteration one unrouted customer into the current partial route. The insertions are performed according to some specific criteria involving minimum additional distance, or maximum savings. A parallel variant of Solomon's sequential insertion procedure is proposed by Potvin and Rousseau (1993). Van Landeghem (1988) extends the savings heuristic by Clarke and Wright (1964) for the VRPTW by also taking into account how good a link between customers is in terms of timing. Bramel and Simchi-Levi (1996) propose an asymptotically optimal heuristic based on the idea of solving the capacitated location problem with time windows.

Most of the recently published VRPTW heuristics use two-phase approaches. In the first phase, a construction heuristic is used to generate a feasible initial solution. During the second phase, an improvement heuristic is applied to the initial solution. These route improvement methods modify the current solution iteratively by performing local searches for better neighboring solutions. Generally, a neighborhood comprises the set of solutions that can be reached from the present one by swapping a subset of *k* arcs. Early route improvement procedures for the VRPTW are proposed by Russell (1977), Baker and Schaffer (1986) and Solomon et al. (1988), who propose a very efficient intra-route improvement procedure based on the Or-opt procedure by Or (1976).

Another Or-opt based method has been suggested by Thompson and Psaraftis (1993). They define the neighborhood of the current solution in terms of feasible transfers of sets of demands belonging to adjacent customers. The exchanges are attempted among a subset of routes that form a cyclic permutation. Russell (1995) develops a hybrid heuristic that invokes the improvement procedure periodically during route construction. Kontoravdis and Bard (1995) describe parallel greedy randomized adaptive search procedure that combines a greedy heuristic and randomization to construct a feasible solution. Local search is then used to improve upon this solution. Other local search implementations can be found for example in Potvin and Rousseau (1995), Antes and Derigs (1995), Shaw (1997, 1998), Caseau and Laburthe (1999) and Cordone and Wolfler-Calvo (2001).

To escape from local optima, the improvement procedures can be embedded in a metaheuristic such as simulated annealing, tabu search or a genetic algorithm. Genetic algorithms belong to the classical local search framework where improvement is sought with each move in the neighborhood of the current solution. In contrast, tabu search and simulated annealing are part of a new paradigm that allows the selection of worse solutions once a local optimum has been reached.

Potvin et al. (1996a) describe a standard tabu search heuristic based on specialized local searches proposed in Potvin and Rousseau (1995). Rochat and Taillard (1995) present a probabilistic technique that uses adaptive memory to record the best routes produced during the search. Also Taillard et al. (1997) and Badeau et al. (1997) use adaptive memory, but they employ a different neighborhood structure, based on exchanges of consecutive customers (or segments) between routes.

De Backer et al. (2000) test four iterative improvement techniques (2-opt, relocate, exchange and cross) within a constraint-programming framework. The techniques are coupled with two metaheuristics (a simple tabu search and guided local search) to avoid the search being trapped in local minima. Guided local search is a metaheuristic based on penalties. The method works by adding a penalty factor to the objective function based on the experience the search has gained. For details, see Voudouris (1997) and Voudouris and Tsang (1998). Kilby et al. (1999) report similar research, but without tabu search and constraint programming. Schulze and Fahle (1999) use a special shift-sequence neighborhood based on the ejection chains of Glover (1991, 1992) within the parallel tabu search framework. Brandão (1999) and Cordeau et al. (2001) introduce simple tabu searches that allow infeasible solutions during the search process. Other successful tabu search implementations can be found in Garcia et al. (1994), Barnes and Carlton (1995), Carlton (1995), Chiang and Russell (1997) and Tan et al. (2000).

In the genetic algorithm proposed by Blanton and Wainwright (1993) the search is driven toward suitable orderings of customers based upon precedence relationships (temporal, spatial, mixed) as well as a fixed a priori global precedence order defined over customer time window lower bounds. Thangiah (1995) uses genetic algorithm to find good clusters of customers, within a "cluster first, route second" problem-solving strategy. Thangiah et al. (1995) test the same approach to solve vehicle routing problems with time deadlines. Thangiah et al. (1994) develop a hybrid method in which the initial solution produced by the genetic algorithm is improved using λ–exchanges. Simulated annealing with a non-monotonic cooling schedule is used to guide the local search and finally tabu search is used to maintain a list of candidate solutions.

In the algorithm proposed by Potvin and Bengio, (1996) new offspring are created by connecting two route segments from two parent solutions or by replacing the route of the second parent-solution by the route of the first parent-solution. Mutation is then used to reduce the number of routes and to locally optimize the solution. Berger et al. (1998) present a hybrid genetic algorithm based on removing certain customers from their routes and then rescheduling them with well-known route-construction heuristics. The mutation operators are aimed at reducing the number of routes by rescheduling some customers and at locally reordering customers. Bräysy (1999a, 1999b) continues the study by Berger et al. (1998) by creating new crossover and mutation operators and by testing the significance of initial solutions.

Homberger and Gehring (1999) propose two evolutionary metaheuristics based on the class of evolutionary algorithms called Evolution Strategies and three well-known route improvement procedures by Or (1976), Osman (1993) and Potvin and Rousseau (1995). Gehring and Homberger (1999 and 2001) use a similar approach with parallel tabu search implementation. Bräysy et al. (2000) describe a two-phase evolutionary algorithm based on hybridization of a genetic algorithm and an evolutionary algorithm consisting of several local search and route construction heuristics. The genetic algorithm used is based on the studies of Berger et al. (1998) and Bräysy (1999a). The recent genetic algorithm by Tan et al. (2001) is based on Solomon's insertion heuristic, λ-interchanges and the well-known PMX-crossover operator.

Bachem, Hochstättler and Malich (1996) use the concept of simulated trading, where the main idea is to use mechanisms of trading in customer assignments. Potvin and Robillard (1995) study a competitive neural network to select seed customers within the parallel construction heuristic of Potvin and Rousseau (1993). Potvin et al. (1996b) examined a similar approach, but determined the parameter values for the construction heuristic with a genetic algorithm. Chiang and Russell (1996) use simulated annealing to guide the hybrid heuristic by Russell (1995). Liu and Shen (1999) develop a new route-neighborhood-based metaheuristic that constructs routes in a nested parallel manner. Gambardella et al. (1999) test a technique based on multiple colonies of artificial ants. Rousseau et al. (2000) use a variable neighborhood descent scheme introduced by Mladenovic and Hansen (1997) and new large neighborhood operators within a constraint-programming framework.

The main contribution of this paper is the development of a new metaheuristic for VRPTW. The proposed method is shown to be currently the most efficient and robust for the VRPTW. The remainder of this paper is organized as follows. An overview of the solution strategy is given first, and then, the different components of the procedure are described in Section 2. First, the construction heuristic used to create initial solutions is described. Second, the new robust ejection chain method is introduced. The basic local search techniques used in minimizing distance are then depicted. Computational experiments assessing the value of the proposed approach is presented in Section 3. Accordingly, a comparative performance analysis involving various metaheuristics is briefly reported. Finally, in Section 4 conclusions are drawn.

2. The Problem Solving Methodology

We propose a new four-phase approach for solving vehicle routing problems. In the first phase several initial solutions are created using a construction heuristic with different combinations of parameter values. In the second phase an effort is made to reduce the number of routes using a new ejection chain-based approach. Note that the routes to be eliminated are sought only in the second phase – in later phases the only objective is to minimize the total traveled distance. In the third and fourth phases we use a new type of particular Variable Neighborhood Search (VNS) technique called Variable Neighborhood Descent (VND), originally proposed by Mladenovic and Hansen (1997).

Contrary to the other metaheuristics based on local search methods, VNS does not follow a trajectory but explores increasingly distant neighborhoods of the current solution, and jumps from this solution to a new one, if and only if an improvement has been made. In this way, favorable characteristics of the current solution (e.g., many variables are already at their optimal value), will often be kept and used to obtain promising neighboring solutions. Moreover, a local search routine is applied repeatedly to get from these neighboring solutions to local optima.

Compared to the VND scheme described in Hansen and Mladenovic (2000), there are three major differences in our implementation. First, instead of a best-accept strategy, we use a first-accept strategy within our improvement procedures. Other differences involve modifying parameter values and objective functions used by the improvement procedures. The proposed scheme is deterministic and one procedure is applied until no improvement can be found. Each time, after all local search operators have been considered once, the values of a set of parameters limiting the search space are increased to perform a more thorough search. Moreover, if none of the chosen local searches is able to improve the solution, we change the objective function to escape from the local minimum. The modification of the objective function is repeated $\bar{h}$ times and it can be done for example by replacing the current objective with a completely new one or by considering simultaneously some alternative objectives and by adjusting the weights of the different objectives. We use the latter approach, where we repeat the alternate objective function for $\bar{p}$ iterations, and after each iteration we modify the weights of the alternate objectives. The scheme exploits the information gathered during the search in a fashion similar to the reactive tabu search heuristic (Battiti and Tecchiolli, 1994) and thus we call our metaheuristic Reactive Variable Neighborhood Search (RVNS). To our knowledge this kind of procedure has not been used before. The proposed metaheuristic can be presented as follows:

```
Step 1. Repeat steps 2 and 3 using all the parameter values within the specified limits. Store the created solutions.
Step 2. Use sequential insertion heuristic to create an initial solution
Step 3. Repeat route elimination procedure until no more routes can be eliminated
Step 4. Identify all the created solutions with the smallest number of routes and insert them into set RB.
Step 5. Repeat step 6 for all $S_i$ in RB and update the best solution found, $S_b$, if needed
Step 6. Reorder the routes in solution $S_i$ according to parameter $\beta$ and improve $S_i$ using the VND procedure.
Step 7. Improve $S_b$ using the post-optimization procedure and return $S_b$.
```

The number of solutions generated in step 1 depends on which of the three parameter sets, specified later in section 3.1, is used. We try 168, 392 and 35 different parameter combinations in sets RVNS(1), RVNS(2) and RVNS(3) respectively. On the other hand, the number of solutions retained in step 4 depends heavily on the problem in question. In some cases, all solutions are retained and sometimes the elimination procedure finds the smallest number of routes only a few times. The reordering of the routes in step 5 is based on the idea of first improving routes having some bad features or weaknesses. Here we consider long waiting time and long average distance with respect to the number of customers on the route as bad features. Parameter β is used to control the weight of these two factors. The larger the value of β the more distance is emphasized in the reordering. Our VND scheme oscillates between two local search operators designed for exchanges of customers between a pair of routes and two operators that perform intra-route improvements. These operators are considered in non-decreasing order of their complexity. If a pair of routes has already been examined and no improvements could be found, and if none of the other operators has modified the routes either, then we avoid double work by disregarding these pairs of routes (or single routes in the case of intra-route improvement operators). Otherwise, if a route is improved by any of the operators, then an attempt is made to improve this route with all pairs of neighboring routes and all operators. We define a pair of routes to be neighbors if the distance between any two customers originally served by separate routes is shorter than a user-defined constant, $\bar{d}$. Moreover, the push-forward and push-backward strategies (Solomon et al., 1988) are used to speed up the feasibility checks of each insertion or move. To reduce the computational effort, we only apply the modification of the objective function at the end of the search to the best solution we have found (fourth phase). Thus, we call the corresponding VND scheme a post-optimization procedure. During this post-optimization, we use waiting time as an alternative objective in addition to total distance.

2.1. Creation of the Initial Solution

The cheapest-insertion-based heuristics used to create the initial solutions borrows 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. The basic idea of Or-opt exchanges is to relocate a chain of consecutive customers. This is achieved by replacing three edges in the original tour by three new ones without modifying the orientation of the route. Here k is a constant parameter value decided by the user.

We tried three basic types of strategies to determine the seed customers. In all strategies, we first select four primary customer nodes (*PC*). The first node is the farthest customer from the depot and the other nodes are determined by finding a node that is geographically as far as possible from the previously chosen customer nodes and depot. Then four secondary customer nodes (*SC*) are selected using the primary nodes. More precisely, since the chosen primary customers form a quadrangle, the secondary customers are the ones closest to the midpoint of each side of the quadrangle. Finally, a set *T* of $\bar{n}$ customers farthest from the depot are selected.

Once the sets *PC*, *SC* and *T* are identified, we select one customer either from set *PC* or *SC* as a starting point to initialize the first route. The next seed customer initializing the next route will be selected from the set $I = T \cup PC \cup SC$. Three different schemes were tested: we first select an unrouted customer in a clockwise or counter-clockwise sweep through the customers in *I* with respect to the previously selected seed. The third scheme is to advance regionally, i.e., select the next seed to be the closest unrouted customer to the first selected seed. Finally, if no unrouted customers can be found in *I*, the next seed customer is the unrouted customer with the lowest index (the customers are indexed from 1 to *n*, where *n* is the total number of customers).

Once the first customer is selected, the unrouted customers are examined one by one and the customer that minimizes the weighted combination of additional detour and waiting time is selected and inserted into the best feasible insertion place. Here, we do not consider all customers for insertion. Instead, we only consider customers that are geographically close to at least one of the previously inserted customers on the route. We consider a customer to be geographically close if the distance from the customer to any customer previously inserted in the current route is shorter than a constant $\bar{d}$ determined by the user. Thus, after each insertion we must update the set of close customers to the current partial route.

The customers farthest away from the depot are usually the most difficult ones to route, since there are often only a few feasible insertion places available for them. If these customers are not considered in the early phase of the solution construction, then in the end we are often forced to create separate routes for them. To avoid this, we favor these remote customers by subtracting from the insertion cost the distance of the corresponding customer to the depot multiplied by a user defined parameter, α_3. More formally, the cost function for customer c_u is given by

$$C_u = \alpha_1 \times D_u + \alpha_2 \times W_u - \alpha_3 \times d_{0u}, \tag{1}$$

where

$$D_u = d_{iu} + d_{uj} - d_{ij},$$

$$W_u = W_u^a - W_u^b,$$

$$\alpha_1 + \alpha_2 = 1, \alpha_3 > 0.$$

Notations d_{iu}, d_{uj} and d_{ij} refer to the distance between the corresponding pair of customers (c_i, c_u), (c_u, c_j) and (c_i, c_j) and W_u^b and W_u^a correspond to the total waiting time before and after the insertion, respectively. Finally d_{0u} is the distance from the customer c_u to the depot and α_1, α_2 and α_3 are parameter values determined by the user. After every k successful insertions, the customers within the route are reordered using a modified Or-opt improvement procedure that uses cost function (1) with α_3=0 to evaluate moves.

2.2. Route Elimination Procedure

We describe here the procedure used in the second phase of our approach. This procedure is used solely to reduce the number of routes. The basic idea is to use a new type of neighborhood structure based on Ejection Chains (EC) proposed by Glover (1991, 1992). The idea of ejection chains is not new in the VRPTW context. They have already been used before, see for example Rousseau et al. (2000) and Caseau and Laburthe (1999). In the context of VRPTW the basic idea is to first remove some customer c_i from route r_k and then to insert some other customer c_j currently served by route r_l into the partial route r_k. If the insertion of c_j is possible, an attempt is made to insert customer c_i in another route $r_m \neq r_k$. If a feasible insertion place can be found, the chain is completed and another customer c_{j+1} is selected to initialize another chain. In each phase within the ejection chain, one customer remains unrouted. The removal and insertion procedures are repeated until we can insert a customer into a neighboring route without the need to remove (eject) any customer. The ejection chain is illustrated in Figure 1.

It is reasonable to assume that the order of the customers on the routes affects the number of feasible insertion places. Therefore, we suggest that a reordering procedure should be attached to the ejection chain procedure. All routes are considered for elimination. However, since it is computationally easier to eliminate shorter routes, we first consider routes having least time-constrained customers for elimination. More precisely, we can calculate the width of the time window by subtracting the value of the earliest time window from the value of the latest time window. We compare the width of each time window to the width of the depot's time window, and if the difference in the widths is over 50%, we consider the particular customer time-constrained. Moreover, since it is computationally prohibitive to try to eliminate a long route having for example 30 customers, only the shortest route is considered for elimination in case of problems with many (over 15) customers per route.

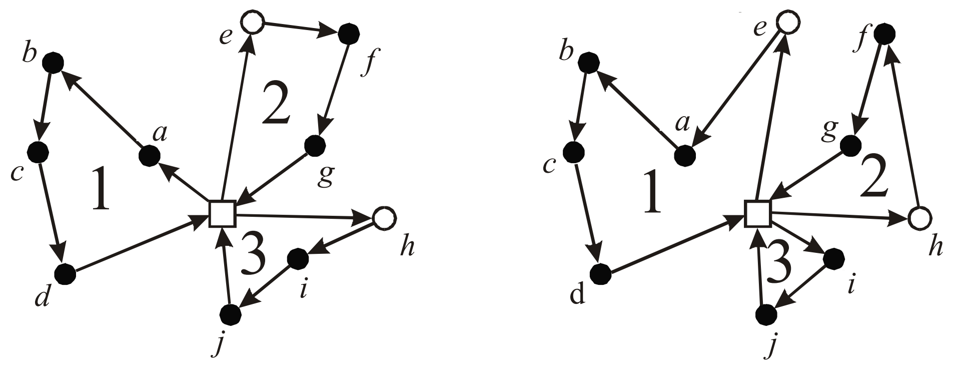

Figure 1: Ejection chain. Let us assume that route 3 is considered for elimination and we try to insert customer *h* denoted by a white circle into route 2. Because of the capacity and/or time constraints, direct insertion is not possible. Thus, we consider reinsertion of the customers on route 2 into other routes than route 3 under elimination. Let us again assume that customer *e* denoted by a white circle can be feasibly inserted into route 1 before customer *a* and that if *e* is first removed from route 2, then we can insert *h* at the beginning of route 2. These insertions are performed in the right picture, thus completing the ejection chain.

We first try to insert each customer on route r_e, currently considered for elimination, directly into some other neighboring route $r_n \in N(r_e)$. A route is considered to be neighboring if it contains customers that are geographically close to a customer of the current route. The limit for closeness $\bar{d}$ is the same as the one used in the creation of the initial solution. Formally

$$N(r_e) = \{r_j \in I^r \mid \min_{c_k \in r_j, c_p \in r_e} \{d(c_k, c_p)\} \leq \bar{d}\}.$$

The notation I^r refers to the set of all routes. The set of neighboring routes is determined before the ejection chain procedure starts, separately for each route of the solution. In addition, we create beforehand *n* different non-decreasing orders for the routes corresponding to the insertion cost of each customer in all routes. The insertion cost is calculated using function (1) with α_1=1 and α_3=0. Thus, in each phase, we first try to insert the customers into routes that are geographically the closest.

If simple insertion fails, a special type of insertion called IR-insert, intelligent reordering, is tried. First, the customer c_i originally served by route r_e is inserted at a location that least increases the value of cost function (1) with α_1=0.5 and α_3=0, and without considering time window constraints. However, the insertion place for customer c_i must be such that we arrive at customer c_i in time. In case the vehicle capacity is exceeded, the insertion cost is set to infinite.

Next, we identify the first customer c_v whose time window constraints are violated. Roughly speaking, there are two alternative ways to reorder the route and thus try to get the route feasible without removing any customer. The simplest way is to consider customers $\{c_1,\ldots,c_{v-1}\}$, i.e., the customers served before customer c_v on the route and try to serve them after c_v on the route, i.e., consider insertion places between any pairs of consecutive customers $\{c_v, c_{v+1}\}$, $\{c_{v+1}, c_{v+2}\},\ldots,\{c_n, c_0\}$, where n is the number of customers on the target route. In this way we arrive earlier at customer c_v, assuming that the triangle inequality holds.

The other possibility is to reorder the customers served before customer c_v so that the duration of the partial route $\{c_1,\ldots,c_{v-1}\}$ is minimized. In other words, the objective is to minimize the time of arrival at customer c_v. These two reordering strategies are considered in turn until either the route is feasible or no feasible insertion places can be found or a user-defined number ($\overline{m}$) of reinsertions has been performed. The procedure is repeated for all the customers for whom the vehicle arrives too late and we accept each move that reduces the lateness, i.e., we use the first-accept strategy within the IR-insert.

If none of the routes can be eliminated directly using simple insertions and IR-insert, we start again from the beginning, restore previous successful direct insertions and apply the ejection chain procedure. Here we use the breadth-first search-strategy. Thus, we first examine all possible chains that require just one ejection and two successful insertions. Then, all possible chains requiring two ejections and three insertions are examined and so on, until the predefined stopping criteria are met.

Let us assume that route r_e is considered for elimination and that IR-insert failed to find a feasible insertion place for customer $c_i \in r_e$. We consider all the neighboring routes $N(r_e)$ one at a time and remove one customer c_d from route r_n $(c_d \in r_n)$, where $r_n \in N(r_e)$. All customers are considered for removal in their original service order. In the next step, an attempt is made to insert c_i into the route r_n using simple insertion followed by IR-insert. If the insertion is successful, then we try to insert c_d into some neighboring route $r_n^{'}$ to route r_n. If this insertion is also successful then the chain is complete and we select another customer c_{i+1} from route r_e. Otherwise, if customer c_d cannot be inserted in any route then we store route r_n in a matrix, reverse the changes related to the last successful insertion and ejection (removal) and select another customer c_{d+1} from route r_n. Or, in case c_d was the last customer on route r_n, we select the first customer on route r_{n+1}, $r_{n+1} \in N(r_e)$.

If we are not able to eliminate a route, special post-processing is applied. In this case, we insert as many customers as possible from the close routes $r_n \in N(r_e)$ into the route r_e just examined to facilitate eliminating some other route. Since we must keep route r_e, it is reasonable to utilize its resources as well as possible. To restrict the number of chains to explore, we set a limit $\bar{l}$ to the allowed increase in distance. Thus if a certain insertion increases the distance of the target route more than this pre-defined limit, the corresponding insertion and/or chain is ignored. Moreover, the maximum length of ejection chains is restricted to $\bar{c}$.

2.3. Route Improvement Procedures

Our VND implementation oscillates between four new local search operators described in detail below. Two of these operators, ICROSS and IRP, are designed for exchanges of customers between a pair of routes and two of the designed operators, IOPT and O-opt, perform intra-route improvements. We accept the first improvement we find and use a new twin variable neighborhood structure: in addition to varying neighborhood structures, the parameter values used by the improvement heuristics are also modified after each successful cycle over all operators. These parameter values are modified so that the neighborhood size (maximum segment length, $\bar{s}$) considered by the procedures increases after every cycle until a user-defined upper-limit is reached. Moreover, within the fourth phase all four improvement operators are considered $\bar{p}$ times and each time the value of parameter α_2 in cost function (1) is increased by $\bar{w}$ units.

ICROSS operator is an extension of the CROSS-exchanges of Taillard et al. (1997). The basic idea of CROSS-exchanges is to first remove two edges (X_1, X_1'), and (Y_1, Y_1') from the first route while the edges (X_2, X_2') and (Y_2, Y_2') are removed from the second route. Then the segments $X_1' - Y_1$ and $X_2' - Y_2$ are swapped by introducing the new edges (X_1, X_2'), (Y_2, Y_1'), (X_2, X_1') and (Y_1, Y_2'). The basic property of CROSS-exchanges is that they preserve the direction of the customers in the selected routes. However, in the case of loose time windows the direction of a segment of consecutive customers may be reversed without causing violations to the time window constraints. In ICROSS we consider both the original and the reversed order for the customers within the segments and select the best one. The reversal could also be optional and controlled by the VND scheme. The maximum segment lengths are managed by the VND scheme, as described above. Figure 2 illustrates this operator.

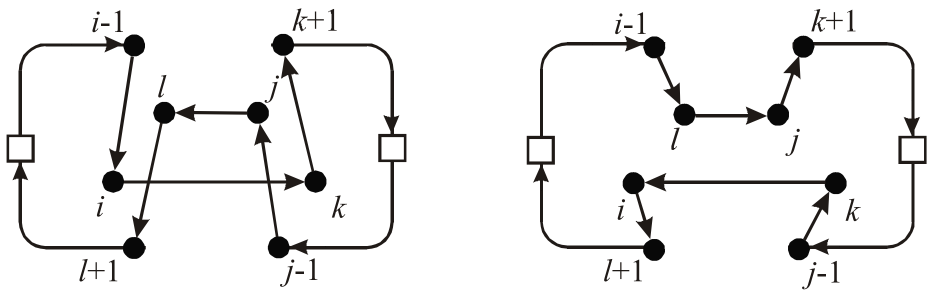

Figure 2: ICROSS-operator. Segments (*i*, *k*) on the upper route and (*j*, *l*) on the lower route are simultaneously reinserted into lower and upper routes respectively. In addition, the order of the customers within both segments is reversed.

Another modification deals with the insertion places considered for each segment. Only insertion positions that are compatible with respect to the time windows of the first and last customer in the current segment, are considered. More precisely, let us assume that we consider insertion between two consecutive customers i and j, and the first and last customers in the current segment are k and l. The cost and feasibility of the insertion are computed only if the earliest allowed start of service at i is smaller than the deadline at l, and the deadline at j is greater than the earliest allowed start of service at k. In addition, we set a fixed limit for the examined insertion places. We consider only the positions within an l-customer neighborhood from the start of the segment in the first route as insertion positions in the second route. The feasibility of the moves is considered only if the corresponding routes are better than the initial routes regarding distance. Finally, if some unfeasibility is found, further feasibility checks are disregarded.

The basic idea in IRP, insert related parallel, is to first remove a set of related customers in a similar manner as in the LNS procedure of Shaw (1997 and 1998). Here, we restrict ourselves to two routes and related customers are selected in a deterministic way. More precisely, we consider customers to be related if the distance between a pair of customers originally served by different routes is within a specified limit, $\bar{d}$. Once a set of related customers is identified and removed, the partial routes are rebuilt using a parallel cheapest insertion heuristic. In each phase we consider all customers and all possible insertion places within the two routes and insert the unrouted customer with the smallest value of cost function (1). Before evaluating the cost value of the constructed routes, the customers on both new routes are reordered using the IOPT operator.

The IOPT operator is a generalization of the Or-opt heuristic. Two major modifications are introduced. First, we consider segments of any length. The maximum segment lengths are managed by the VND scheme, as described above. The other modification deals with trying to reverse the order of the customers within the selected segment. IOPT considers both the original and reversed order for the customers in the current segment and selects the move that yields a better output. This inversion could also be optional and controlled by the VND scheme.

It is reasonable to assume that in sequential insertion heuristics the previously inserted customers impact the selection and order of the subsequently inserted customers and thus solution quality. The O-opt operator tries to deal with this problem by first selecting $\bar{i}$ customers so that they are geographically as dispersed as possible. More precisely, we first find the customer on the current route that is the most distant from the depot. Then we select $\bar{i}-1$ customers sequentially so that they are as far as possible from the previously selected customers. Next the $\bar{i}$ customers are put in all feasible orders and each of the partial routes initialized by these $\bar{i}$ customers is rebuilt with the cheapest insertion heuristics. Thus, the basic idea is to use more than one customer to initialize the routes. To further reduce the computational efforts, we disregard partial routes with $\bar{e}$ % higher distance compared to the initial route. The cheapest insertion heuristic inserts the customers in increasing order of their time window width. Finally, after $\bar{k}$ customers have been inserted, the route is reordered using the Or-opt operator.

3. Computational Comparison of the Procedures

3.1. Problem Data and Parameter Values

Our metaheuristic was tested on four different data sets taken from Solomon (1987), Gehring and Homberger (1999) and Russell (1995). The first consists of six sets (R1, C1, RC1, R2, C2, RC2), each of which contains between eight and twelve 100-node problems over a service area defined on a 100 × 100 grid. For R1 and R2, the customer locations are distributed uniformly over the service area. Sets C1 and C2 have clustered customers, and sets RC1 and RC2 have a combination of clustered and randomly located customers. In addition, R1, C1 and RC1 have tight time windows and a vehicle capacity of 200 units; R2, C2, and RC2 have a long scheduling horizon and vehicle capacity of 1000, 700, and 1000 units, respectively. In all, there are 56 problem instances. The time window and the vehicle capacity constraints in problem sets R1, C1, and RC1 allow only a small number of customers to be served by each vehicle. The opposite is true for R2, C2, and RC2.

To test scaling issues, we solved larger problem sets of 200 and 400 customers as reported by Gehring and Homberger (1999). These problem sets are created in a similar manner to Solomon's corresponding problem set and there are 6 sets and 10 problems in each set. Gehring and Homberger (1999) created 300 extended problems using the principles of Solomon. The number of customers varies between 200 and 1000 in the problems. Finally, we also solved real-life problems D417 and E417 as reported by Russell (1995). The problems are extracted from a fast food routing application in the South Eastern United States. There are 417 customers in both problems and the problems differ only in that the problem E417 has a higher percentage of tight time windows. The method was coded in JAVA, and run on a Pentium 200 MHz personal computer.

In some cases, the parameter values used depend on the characteristics of the problem. It is indicated to use a different parameter set in case of larger and more complex problems to reduce computational effort, but in addition to that, we also consider the average number of customers on the routes. We have two reasons for doing this. In the case of routes with many customers, the route often covers a larger geographical area and customers have wider time windows. This makes possible to consider geographically more distant customers for insertion (larger value for $\bar{d}$). On the other hand, in the case of longer routes, the number of routes is generally smaller and there are often much more insertion places available for customers, thereby increasing the complexity of the search. Therefore, shorter ejection chains are used in the problems with many customers per route. The identification of the problem type is performed by summing up the demands of all customers and by dividing it by the vehicle capacity. In this way, we get an estimate for the number of routes required and for the average number of customers per route. We divide the problems into two groups. To identify the groups, we set a limit at 25 for the average number of customers.

We found it computationally intractable to optimize the value of each parameter separately. Therefore, we tuned the parameter values only once by trying to find a "local minimum" in the following way. First, we selected a parameter setting based on intuition. Then, several values were tried for each parameter, while keeping the other parameter values fixed. Here, the order in which the parameters are considered is important. We first analyzed the effect of each parameter separately and hence determined the value of the least sensitive parameters. To reduce the workload, only a set of four test problems and a few (3-10) intuitively selected values were tried. Each time, we selected a parameter value that gave the best average output for the selected four test problems. Because of limited computational resources we tried to select the parameter values and number of parameter combinations so that the average run time would be less than one hour on a 200 MHz PC. The parameters and their values are: $\bar{d}=20$ and $\bar{c}=9$ in group I, $\bar{d}=30$ and $\bar{c}=4$ in group II, and $\bar{d}=10$ within IRP, $\bar{h}=1$, $\bar{s}=3–5$, $\bar{p}=3$, $\bar{m}=5$, $\bar{n}=30$, $\bar{w}=0.2$, $\beta=0.5$, $\bar{k}=10$, $k=3$, $\bar{i}=4$, $l=15$, $\bar{l}=15\%$, $\bar{e}=30\%$.

Most of the parameters described above are used to limit the search space, and thus larger values often (not necessarily) yield better output. The only exceptions are $\bar{w}$ and β. One can separate several groups of interdependent parameters. First, $\bar{d}$, $\bar{m}$, $\bar{l}$ and $\bar{c}$ are used to control the complexity of the route elimination procedure and their values should therefore be determined simultaneously, based on the computational resources available. Also $\bar{h}$, $\bar{p}$ and $\bar{w}$ controlling the post-optimization scheme, $\bar{s}$ and l used by ICROSS and $\bar{k}$, $\bar{i}$ and $\bar{e}$ within O-opt are interdependent.

The reader must note that even if there are a lot of different parameters, we used a fixed value for all the parameters above during the computational experiments. This is due to the fact that in most cases the results are not sensitive to changes in parameter values. The effect of the above parameter values on total distance was less than 1%. The only exceptions were $\bar{s}$ and k, where the difference in total distance between the best and worst values varied between 3–7%. However, in most cases the best values were the same and we were able to find a single robust value. Parameters α_1, α_3 and the seed selection schemes clearly had a bigger impact on the results. For example, the difference between various seed selection schemes in terms of total distance was in some cases greater than 50%. We found it impossible to determine a robust value for them, which would give good results for all test problems. Therefore, we decided to try several values for these parameters within specified ranges, as depicted below. Since these three parameters are interdependent, we determined the bounds for them simultaneously by trying 16 different seed selection schemes, described in detail in Bräysy (2001), and all values for α_1 and α_3 within ranges 0.1–1.0 and 0–3, respectively (in increments of 0.1 units). More precisely, we considered one parameter at a time, while varying the other two parameters within the given ranges. In this manner we could identify the values that give the best output and determine the bounds accordingly by considering also the total number of combinations tried. The parameter sensitivity is discussed in more detail in Bräysy (2001).

We set the following bounds for the most crucial parameters: α_1: 0.7–1.0 (in increments of 0.1 units), α_3: 0.5–1.7 (in increments of 0.2 units). Within the VND scheme α_1 is always 1, except in the O-opt operator, where it is 0.9. The number of selection schemes tried was set to 6. Within the post-optimization, the distance of the routes is allowed to increase by 10% if the value of the cost function is improved. Finally we set a maximum limit of 300 to the ejection chains stored in memory in order to keep the memory requirements reasonable. We denote the basic parameter set described above by RVNS(1). In addition, we created another set, RVNS(2), by increasing the number of selection schemes to 14. Thus the only difference between RVNS(1) and RVNS(2) is the number of seed selection schemes considered. For the larger problem sets we selected the following parameter values based on intuition: α_1: 0.6−1.0 (in increments of 0.1 units), α_3: 0.5−1.7 (in increments of 0.2 units), $\bar{n}$=100, l=35, and the number of seed selection schemes is one. All other parameter values are the same as in the set RVNS(1). We denote the parameter set used to solve large problem sets by RVNS(3). The right value for parameter $\bar{n}$ is strongly dependent on the problem size, so instead of using two different values 35 and 100 for 100-customer problems and others respectively, one should relate the value directly to problem size. According to our experience, 30% of the total number of customers works relatively well.

Given that the procedure used to reduce the number of routes is powerful and often produces the minimum number of routes, we used the following heuristic rule to decrease the computational workload. If we have previously obtained a solution with a smaller number of routes than now, or the converse, we use the lower value as an estimate for the minimum number of routes. This information is then used to stop the route elimination procedure once a number of routes equals to the estimate is found, without spending heavy computational efforts in trying to eliminate additional routes. In case the elimination procedure returns the same number of routes each time, we compare the values to the number of routes in the initial solutions. If the values are the same, then we assume that the problem is easy and we have obtained the minimum number of routes. Otherwise, the length of ejection chains is increased to $\bar{c}$=10, the geographical closeness range is increased to $\bar{d}$=40 and $\bar{l}$=65%, and the elimination procedure is repeated in order to introduce more power to the search.

3.2. Experimental Analysis of the Proposed Procedures

The performance of the ICROSS and IRP-procedures is compared with the well-known inter-route improvement heuristics in Table 1. The heuristics used in the comparison are CROSS-, 2-opt*, λ-exchange, relocate and exchange-operators proposed by Taillard et al. (1997), Potvin and Rousseau (1995), Osman (1993) and Savelsbergh (1992), respectively. Due to limited computational resources, we only use a subset of six problems in Tables 1 and 2 below. This subset was created by selecting one problem from each of the six problem groups of Solomon (1987). We tried to select the problems so that they cover different characteristics and are not the easiest instances to solve.

Table 1: A comparison of inter-route improvement heuristics. TIME stands for the time consumption in seconds for independently improving the initial solutions 50 times. Moreover, the value of total distance obtained with each operator is given.

Method:	TIME	R105	RC101	RC206	R202	C103	C204
IRP	12	1576.2	1838.5	1313.4	1515.0	839.6	618.9
ICROSS	16	1425.0	1735.6	1345.4	1340.1	874.8	622.8
CROSS	12	1446.6	1735.6	1345.4	1356.6	884.5	622.8
λ-EXCHANGE	6	1519.2	1760.4	1345.4	1376.1	922.0	622.8
RELOCATE	4	1546.9	1853.8	1348.5	1382.7	1160.1	622.8
EXCHANGE	1	1596.7	1784.2	1437.5	1512.4	1145.9	623.3
2-OPT*	1	1551.6	1781.0	1422.5	1475.3	1142.4	623.3

According to Table 1 none of the described operators is able to dominate all others in all 6 test problems. However, in each case the best results are obtained either by using ICROSS or IRP, and in general ICROSS appears to perform best. One of the greatest advantages of the ICROSS is its robustness – it performs well in all tested problems. Similar findings were also obtained in comparison of different intra-route local searches: our IOPT and O-opt outperformed Simple insertion and Or-opt, though the differences remained nonsignificant.

Since our operators are more sophisticated than the previously introduced approaches, it is reasonable to assume that the time consumption is greater as well. For example, ICROSS is a generalization of the CROSS, λ-exchange (if λ=1), relocate, exchange and 2-opt* operators and thus requires more time, as can be seen from Table 1. In the same way, λ-exchange generalizes relocate and exchange and therefore performs better, as can be seen from Table 1. However, here one must note that since the first-accept strategy is used and previously performed moves always have a cumulative effect on the subsequent moves, it is not always clear that a more complicated operator yields better final results.

By comparing ICROSS and CROSS one can observe that ICROSS produces slightly better solutions on R105, R202 and C103 and equally good solutions on the other three test problems. Thus, one can conclude that even in the case of routing problems with time windows, the reversal of the customer service order often improves the solution quality. Here though, one must bear in mind that the differences are quite moderate, varying between 1.1 and 1.5%. The increase in time consumption appears to be only 33% over the burden of CROSS-exchanges. The reason for this is that the most time-consuming part of ICROSS is the feasibility-checking procedure for each move, and it is performed only if the corresponding routes are improved in terms of distance. On the other hand, the feasibility of the other route with inverted segment is checked only if the first route is found to be feasible and this often reduces the workload.

Another observation is that according to Table 1, IRP has difficulties in achieving robustness. For example, it is not able to improve the initial solution of R202 at all. Still, even if it is the worst on problem R202, it yields the best outcome to quite a similar problem, RC206. One reason for this behavior is that IRP considers only distance when reinserting the related customers, and in the case of routing problems with time windows this is not always the best strategy. Since R202 has tight time constraints, one should consider also temporal aspects. One way to deal with the problem might be to dynamically change the weight of geographical and temporal factors in the cost function according to the features of the problem in question.

The first row in Table 2 illustrates the results obtained by running the whole VND scheme using all four improvement operators while not using the post-optimization procedure. The second row represents the results obtained without using ICROSS, and the third, fourth and fifth rows depict the results obtained without IRP, IOPT and O-opt respectively. The four last rows describe the results obtained by using only one operator (ICROSS, IRP, IOPT or O-opt) and thus disregarding the VND scheme.

Table 2: Comparison of the total traveled distance produced by the whole VND scheme with approaches that omit one of the improvement procedures and with results produced by only a single improvement procedure. The results are the best of 100 combinations of random parameter values.

Method:	R105	RC101	RC206	R202	C103	C204
FULL VND	1394.6	1747.2	1221.1	1207.2	828.1	590.6
VND – ICROSS	1427.2	1805.6	1298.8	1311.4	828.9	619.3
VND – IRP	1396.9	1747.2	1221.2	1228.2	828.1	602.5
VND – IOPT	1394.6	1747.2	1223.6	1207.2	828.1	590.6
VND – O-OPT	1394.6	1747.2	1221.1	1215.1	828.1	594.1
ICROSS	1410.6	1756.4	1250.4	1292.2	851.1	671.6
IRP	1427.2	1805.6	1368.1	1326.2	833.8	619.3
IOPT	1486.1	1823.5	1441.4	1405.7	1021.4	619.5
O-OPT	1486.1	1823.5	1441.8	1391.1	1017.4	613.2

According to Table 2 the full VND scheme yields better results to every test problem than any of the single improvement procedure. Regarding the individual operators, ICROSS is clearly the best method. It performs best by itself, and also, the total traveled distance of the whole VND scheme is the worst if ICROSS exchanges are not considered. The two intra-route operators IOPT and O-opt seem to be less significant. Since the heuristic creating the initial solutions reorders the routes using Or-opt exchanges after each *k* customer insertions, the routes of the initial solutions are already well-ordered. Also the influence of the IRP operator seems to be quite small. Here, one must note that using only a single value for parameter α_1 clearly affects the results of IRP.

Table 3 illustrates the solution values obtained for the test problems both with and without using the minima-escaping procedure, i.e., changing the objective function. In the course of the experiments we found that it was too time-consuming to improve each of the created solutions separately by varying the objective function. Therefore, we decided to apply it only to the best solution found during the search. The average time consumption of this post-optimization procedure was approximately 0.8 minutes, which is only about 2% of the total time consumption of the RVNS(1) method, which was 37 minutes.

Table 3: The effect of the post-optimization procedure on the average traveled distance of the final solutions in each of the six problem groups.

	R1	R2	C1	C2	RC1	RC2
No Post-opt	1232.91	995.41	828.38	590.30	1404.52	1147.13
Post-opt	1229.48	989.62	828.38	590.30	1394.26	1141.57

As can be seen from Table 3, the significance of the post-optimization regarding total distance is generally quite small, only about 0.5% on the average. It was not able to improve the results of C1 and C2 at all. Here though, one must note that the RVNS(1) without post-optimization already yields optimal solutions on problems in groups C1 and C2, except on problem C204 in group C2. However, considering the moderate time consumption and the achievement of two new best-known solutions, we found the post-optimization procedure to be valuable. The variation of the objective function seems to help in escaping from the local minimum.

3.3. Comparative Analysis of Algorithms

In Table 4 we compare the results obtained in a single run with our RVNS(1) and RVNS(2) parameter sets with the best results reported in the recent studies. Since only LS, HG1 and RGP report the number of runs and time consumption used to obtain the results in Table 4, we do not consider the computational burden here. All methods in Table 4 consider the number of vehicles as the primary objective and total distance as the secondary objective.

Table 4: Comparison of the final solutions produced by the proposed RVNS method with two different parameter sets, RVNS(1) and RVNS(2), with the results of the best metaheuristics proposed recently by other authors. For each problem group the average number of vehicles and total distance are given. The notation CNV in the last row indicates the cumulative number of vehicles over all 56 test problems.

PROB.	TBGGP	CR	LS	GTA	HG	RGP	CLM	RVNS(1)	RVNS(2)
R1	**12.17**	**12.17**	**12.17**	**12.00**	**11.92**	**12.08**	**12.08**	**12.00**	**11.92**
	1209.35	1204.19	1249.57	1217.73	1228.06	1210.21	1210.14	1229.48	1222.12
R2	**2.82**	**2.73**	**2.82**	**2.73**	**2.73**	**3.00**	**2.73**	**2.73**	**2.73**
	980.27	986.32	1016.58	967.75	969.95	941.08	969.57	989.62	975.12
C1	**10.00**	**10.00**	**10.00**	**10.00**	**10.00**	**10.00**	**10.00**	**10.00**	**10.00**
	828.38	828.38	830.06	828.38	828.38	828.38	828.38	828.38	828.38
C2	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**
	589.86	591.42	591.03	589.86	589.86	589.86	589.86	590.30	589.86
RC1	**11.50**	**11.88**	**11.88**	**11.63**	**11.63**	**11.63**	**11.50**	**11.50**	**11.50**
	1389.22	1397.44	1412.87	1382.42	1392.57	1382.78	1389.78	1394.26	1389.58
RC2	**3.38**	**3.25**	**3.25**	**3.25**	**3.25**	**3.38**	**3.25**	**3.25**	**3.25**
	1117.44	1229.54	1204.87	1129.19	1144.43	1105.22	1134.52	1141.07	1128.38
CNV	**410**	**411**	**412**	**407**	**406**	**412**	**407**	**406**	**405**

The research teams in Table 4 are:
TBGGP: Taillard et al. (1997), CR: Chiang and Russell (1997), LS: Liu and Shen (1999), GTA: Gambardella et al. (1999), HG: Homberger and Gehring (1999), RGP: Rousseau et al. (2000), CLM: Cordeau et al. (2001), RVNS(1) and RVNS(2): this paper

A very important observation regarding Table 4 is that none of the methods is able to dominate all other methods in all problem groups. The purpose of Table 4 is solely to demonstrate that we are competitive with the best results of the best methods, even if a limited amount of parameter values and a single run is used. For clustered problem groups, many methods yield equally good results, but in other problem groups RVNS(2) dominates all previous approaches in at least three cases out of four. The CNV of RVNS(2), is the lowest known at 405. It matches the minimum number of routes reported for the best-known solutions every time, except for problem R101 for which Thangiah et al. (1994) report a solution requiring one route less. Moreover, we were able to find four new best known solutions that are given in Appendix 2.

Table 5: Comparison of the results for real-life problems by Russell (1995) with the results of the best heuristic and metaheuristic methods. The Time column shows the average time consumption of a single run in minutes, and Runs refers to the number of runs required to get the reported results.

Reference	**Time**	**Runs**	**Problem D417**		**Problem E417**	
			Vehicles	**Distance**	**Vehicles**	**Distance**
TOS	26	-	54	4866	55	4149
KB	11	5	55	4273.4	55	4985.7
RT	-	-	54	6264.80	54	7211.83
R	7	3	55	4964	55	6092
CR1	25	-	55	4232.39	55	4397.49
TBGGP	-	-	55	3439.8	55	3707.1
CR2	37	-	55	3455.28	55	3796.61
LS	45	3	54	3747.52	54	4691.14
HG1	30	5	54	4703	55	4732
HG2	30	5	54	9708	54	5174
RVNS(3)	378	1	54	3506.21	54	3801.64

The authors in Table 5 are:
TOS: Thangiah et al. (1994), KB: Kontoravdis and Bard (1995), RT: Rochat and Taillard (1995), R: Russell (1995), CR1: Chiang and Russell (1996), TBGGP: Taillard et al. (1997), CR2: Chiang and Russell (1997), LS: Liu and Shen (1999), HG1: Homberger and Gehring (1999), HG2: Homberger and Gehring (1999), RVNS(3): this paper.

In the end, the performance of the RVNS(3) was tested on the larger problems introduced by Russell (1995) and Gehring and Homberger (1999) to test scaling issues. The results are presented in tables 5–7. Regarding the solution quality, the performance was found to be excellent. For the real-life problems by Russell (1995) we obtained the best-known solutions using just a single trial and without optimizing the parameter values in any way, though our method requires more time than the previous approaches. The reason for the higher time consumption, compared for example to the 400-customer problems by

Gehring and Homberger (1999), is that the distances between customers are clearly smaller in D417 and E417. Thus, there are many more alternatives examined by the algorithms. We should perhaps have used lower values for parameters such as $\bar{d}$ and $\bar{l}$ instead of fixed values in order to reduce the number of insertions considered.

Table 6: Results for 200 customer problem sets by Gehring and Homberger (1999) using parameter set RVNS(3). The time columns depict the time consumption in minutes.

	GEHRING et al. (1999)			BRÄYSY (2001)		
	Vehicles	**Distance**	**Time**	**Vehicles**	**Distance**	**Time**
C1	18.90	2782	40	18.90	2778.80	12
C2	6.00	1846	40	6.00	1842.43	28
R1	18.20	3705	40	18.10	3821.43	21
R2	4.00	3055	40	4.00	3045.29	12
RC1	18.00	3511	40	18.00	3508.07	20
RC2	4.30	2658	40	4.40	2628.36	15

From tables 6 and 7 one can see that our approach dominates Gehring and Homberger (1999) in five out of six sets of 200-customer problems, and four out of six sets of 400-customer problems, though the differences seem to be small, especially in 200-customer problems. The number of routes seems to be the same for both methods. For R1, our approach requires one vehicle less, but on the other hand our method is worse on RC2.

Table 7: Results for 400 customer problem sets by Gehring and Homberger (1999) using parameter set RVNS(3). The Time columns depict the time consumption in minutes.

	GEHRING et al. (1999)			BRÄYSY (2001)		
	Vehicles	**Distance**	**Time**	**Vehicles**	**Distance**	**Time**
C1	38.00	7584	80	38.00	7321.68	70
C2	12.00	3939	80	12.00	3922.71	65
R1	36.30	8925	80	36.20	9154.50	129
R2	8.00	6502	80	8.00	6547.87	118
RC1	36.10	8763	80	36.10	8628.74	133
RC2	8.60	5507	80	8.70	5633.28	78

Regarding time consumption, our method seems to be faster on the 200-customer problems, but slightly slower on the 400-customer problems. However, considering that we implemented our algorithms with JAVA, which is often considered much slower than C, one can conclude that our approach is faster and that its overall performance is better on larger test problems. This is especially due to the fact that we did only one trial without optimizing the parameter values in any way. One must note that in Tables 6 and 7 the time consumption reported by Gehring and Homberger is multiplied by four, since they used a parallel implementation with four processors.

In Table 8 we only included approaches where a sufficient amount of information is provided by the authors. At least, the computer type, the number of computational runs, as well as the time consumption, number of vehicles and total traveled distance for each problem group must be reported. In cases where several results are reported by the authors using different running times, we selected the one that best facilitates the comparison. One must note that the computational times are scaled to equal the running times of a Sun Sparc 10/50 using Dongarra's (1998) factors. In addition, if several runs are required to get the reported results, the computational times are multiplied by this number to see the real computational burden. Details for calculating the values in Table 8 can be found in Bräysy (2001). One must note that the times described in Table 8 are only indicative and should be used solely to get a picture of the magnitude of the running times. The average number of routes and total traveled distance are rounded after the first decimal place and to nearest integer, respectively, to clarify the Table. For the same reason, problem groups C1 and C2 are not considered. Another reason for disregarding them is the easy nature of the clustered problems and thus the often insignificant differences between the various approaches. Most of the approaches in Table 8 use a similar objective function, where the primary objective is to minimize the number of routes and the secondary objective is to minimize either total traveled distance or total time. The only exception is KPS, where the only objective is to minimize total distance.

The first observation from Table 8 is that none of the methods is able to dominate all other methods in all problem groups. Considering the solution quality and time consumption, our approach seems to be most efficient. RVNS(1) and RVNS(2) dominate all other approaches except HG1, HG2 and HG3 in all problem groups. RVNS(2) also dominates HG2 and HG3 in all cases and HG1 in three cases out of four. Moreover, RVNS(1) and RVNS(2) are clearly faster than HG1 and HG2. HG3 appears to be faster than RVNS(1) and RVNS(2), but here one must note that our methods were implemented using JAVA, which is clearly slower than C. The time consumption of our methods would be smaller if C had been used.

Table 8: Comparison of the solution quality and computational burden. The notations NV, TD and TIME indicate the average number of vehicles, average total distance and total time consumption in minutes, respectively, with regard to the four problem groups of Solomon (1987).

PROB	R1			R2			RC1			RC2		
	NV	TD	TIME	NV	TD	TIME	NV	TD	TIME	NV	TD	TIME
S	13.6	1437	<0.1	3.3	1402	<0.1	13.5	1597	<0.1	3.9	1682	<0.1
TP	13.1	1367	<0.1	3.1	1299	<0.1	13.0	1534	<0.1	3.7	1672	<0.1
PR	13.3	1509	<0.1	3.1	1387	<0.1	13.4	1724	<0.1	3.6	1651	<0.1
GPR	12.9	1320	2.5	3.1	1229	1.8	12.9	1483	2.2	3.9	1551	1.8
R	12.7	1317	0.6	2.9	1167	0.9	12.4	1523	0.5	3.4	1398	0.8
KB	12.6	1325	6.1	3.1	1164	9.6	12.6	1501	6.0	3.5	1414	10.6
AD	12.8	1386	0.3	3.1	1367	0.2	12.5	1546	0.2	3.4	1598	0.2
RT	12.6	1197	67.5	3.1	954	245.0	12.4	1370	65.0	3.6	1140	195.0
BHM	12.8	1417	10.2	3.1	1250	6.1	12.5	1532	9.5	3.4	1512	8.3
PKGR[a]	12.6	1295	10.7	3.1	1186	12.0	12.6	1465	9.8	3.4	1476	11.0
TBGGP	12.3	1220	229.6	3.0	1013	337.2	11.9	1381	187.7	3.4	1199	193.3
S 1	12.4	1206	75.0	—	—	—	12.1	1363	75.0	—	—	—
S 2	12.5	1198	94.5	—	—	—	12.1	1361	94.5	—	—	—
C-WC	12.5	1242	41.5	2.9	995.4	40.0	12.4	1409	21.7	3.4	1140	28.4
SF	12.5	1268	218.3	3.1	1056	524.3	12.3	1396	193.9	3.4	1308	397.6
GTA	12.4	1211	210.0	3.0	960.3	210.0	11.9	1388	210.0	3.3	1149	210.0
KPS	12.7	1200	362.5	3.0	966.6	362.5	12.1	1388	362.5	3.4	1133	362.5
LS	12.2	1250	225.8	2.8	1017	33.9	11.9	1413	155.3	3.3	1205	36.3
HG1	11.9	1228	275.0	2.7	967.0	352.0	11.6	1393	242.0	3.3	1144	363.0
HG 2	12.0	1226	431.2	2.7	1034	477.4	11.5	1407	515.9	3.3	1176	469.3
HG 3[b]	12.4	1201	44.0	2.9	945	44.0	12.0	1356	44.0	3.3	1140	44.0
RVNS(1)	12.0	1229	125.7	2.7	990	107.2	11.5	1394	102.2	3.3	1141	51.9
RVNS(2)	11.9	1222	288.1	2.7	975	259.3	11.5	1390	214.6	3.3	1128	129.8

[a] Tabu A- implementation.

[b] Method ES4C using 4 communicating processors.

The authors in Table 8 are:

S: Solomon (1987), TP: Thompson and Psaraftis (1993), PR: Potvin and Rousseau (1993), GRP: Garcia et al. (1994), R: Russell (1995), KB: Kontoravdis and Bard (1995), AD: Antes and Derigs (1995), RT:

Rochat and Taillard (1995), BHM: Bachem et al. (1996), PKGR: Potvin et al. (1996a), TBGGP: Taillard et al. (1997), S1: Shaw (1997), S2: Shaw (1998), C-WC: Cordone and Wolfler-Calvo (2001), SF: Schulze and Fahle (1999), GTA: Gambardella et al. (1999), KPS: Kilby et al. (1999), LS: Liu and Shen (1999), HG1: Homberger and Gehring (1999), HG2: Homberger and Gehring (1999), HG3: Gehring and Homberger (1999), RVNS(1) and RVNS(2): this paper.

According to Table 8, there are several heuristic and metaheuristic methods that are faster than RVNS(1) or RVNS(2). However, all of them are clearly inferior regarding solution quality. For example, the difference between S and RVNS(1) in problem group RC2 is about 18% in the number of vehicles and about 47% in total traveled distance, which are clearly unacceptable from a practical viewpoint.

4. Conclusions

We proposed a new construction heuristic and five new improvement procedures. These were used to design a new metaheuristic for the VRPTW. The proposed method uses a new four-phase approach. In the first phase an initial solution is created using the construction heuristic. Then, a special route-elimination operator, based on a new type of ejection chain is used to minimize the number of routes. In the third and fourth phases the created solutions are improved in terms of distance using a new type of particular Variable Neighborhood Search technique called Variable Neighborhood Descent. In these schemes, we apply cyclically four new types of improvement procedures until no improvement can be found. In addition, after each cycle of application of the four operators, the neighborhoods are enlarged by increasing the segment lengths considered by these operators. In the fourth phase the objective function used by the local search operators is modified to also consider waiting time to escape from a local minimum. The computational testing of the proposed methods was carried out with the 56 test problems of Solomon, 2 real-life problems by Russell (1995) and 120 test problems of 200 and 400 customers taken from Gehring and Homberger (1999).

In the comparison of the two proposed inter-route improvement operators with the other well-known inter-route improvement procedures, it was found that the best outcomes were always obtained with one of the two procedures proposed in this paper. The same finding also applies to our intra-route procedures, although the differences between competing approaches were small. Finally, the variable neighborhood scheme was found to give better output than any of the single operators, and when analyzing the importance of the post-optimization procedure (phase four), it was found that in most cases small improvements can be found quickly. Thus it was concluded that alternating the objective function seems to help in escaping the local minimum.

In the end, different comparisons were performed with the other previous approaches for solving VRPTW. In the first comparison, the results of two different parameter sets, denoted by RVNS(1) and RVNS(2), were compared with the best results obtained with the best metaheuristics, studied by other authors. The purpose of this comparison was mainly to illustrate that our approaches can consistently produce results that are competitive with the best known results.

To make possible a proper comparison between the different methods proposed in the literature, in terms of efficiency, we modified the computation times using Dongarra's (1998) factors. In terms of solution quality, the best performing methods were RVNS(1), RVNS(2) and the methods of Homberger et al. (1999) and Gehring and Homberger (1999). To the best of our knowledge, RVNS(2) is the only approach that can consistently produce the lowest known cumulative number of vehicles (405) to Solomon's test problems. In addition, four new best-known solutions were obtained. The performance of RVNS(3) was found to be excellent in the larger test problems proposed by Russell (1995) and Gehring and Homberger (1999). Due to its deterministic nature, our metaheuristic is more reliable than the best competing approaches. In addition, our method seems to be very robust, yielding feasible solutions of acceptable quality to all types of well-known test problems in every case. We conclude that the approach presented in this paper is currently the most effective one for the vehicle routing problem with time windows.

Future work will explore the usage of more sophisticated schemes for creating initial solutions as well as implementation on parallel computers. Currently, the most time-consuming part of the search is the route elimination procedure. New strategies for speeding it up will be examined and the proposed procedures will also be tested on other routing problems.

Appendix 1

The detailed best results of our approaches to Solomon's test problems. For each problem both number of vehicles and total distance are depicted.

THE RESULTS OBTAINED WITH RVNS(1)					
R1..	**R2..**	**C1..**	**C2..**	**RC1..**	**RC2..**
01/ 19/ 1652.22	**01**/ 4/ 1260.91	**01**/ 10/ 828.94	**01**/ 3/ 591.56	**01**/ 14/ 1698.82	**01**/ 4/ 1428.09
02/ 17/ 1486.22	**02**/ 3/ 1198.56	**02**/ 10/ 828.94	**02**/ 3/ 591.56	**02**/ 12/ 1579.75	**02**/ 3/ 1375.45
03/ 13/ 1311.11	**03**/ 3/ 957.02	**03**/ 10/ 828.06	**03**/ 3/ 591.17	**03**/ 11/ 1280.14	**03**/ 3/ 1062.52
04/ 10/ 999.59	**04**/ 2/ 894.69	**04**/ 10/ 824.78	**04**/ 3/ 594.06	**04**/ 10/ 1143.86	**04**/ 3/ 812.81
05/ 14/ 1381.45	**05**/ 3/ 1032.96	**05**/ 10/ 828.94	**05**/ 3/ 588.88	**05**/ 13/ 1632.34	**05**/ 4/ 1326.83
06/ 12/ 1262.49	**06**/ 3/ 929.62	**06**/ 10/ 828.94	**06**/ 3/ 588.49	**06**/ 11/ 1432.12	**06**/ 3/ 1197.46
07/ 10/ 1155.29	**07**/ 2/ 982.01	**07**/ 10/ 828.94	**07**/ 3/ 588.29	**07**/ 11/ 1234.30	**07**/ 3/ 1071.48
08/ 9/ 974.85	**08**/ 2/ 737.17	**08**/ 10/ 828.94	**08**/ 3/ 588.32	**08**/ 10/ 1152.77	**08**/ 3/ 853.90
09/ 11/ 1238.58	**09**/ 3/ 944.94	**09**/ 10/ 828.94			
10/ 10/ 1132.37	**10**/ 3/ 975.26				
11/ 10/ 1139.44	**11**/ 2/ 972.55				
12/ 9/ 1019.41					
THE RESULTS OBTAINED WITH RVNS(2)					
R1..	**R2..**	**C1..**	**C2..**	**RC1..**	**RC2..**
01/ 19/ 1650.80	**01**/ 4/ 1253.21	**01**/ 10/ 828.94	**01**/ 3/ 591.56	**01**/ 14/ 1697.43	**01**/ 4/ 1417.60
02/ 17/ 1486.12	**02**/ 3/ 1197.03	**02**/ 10/ 828.94	**02**/ 3/ 591.56	**02**/ 12/ 1569.33	**02**/ 3/ 1375.45
03/ 13/ 1299.14	**03**/ 3/ 944.55	**03**/ 10/ 828.06	**03**/ 3/ 591.17	**03**/ 11/ 1274.02	**03**/ 3/ 1062.52
04/ 9/ 1012.76	**04**/ 2/ 894.69	**04**/ 10/ 824.78	**04**/ 3/ 590.60	**04**/ 10/ 1140.51	**04**/ 3/ 799.40
05/ 14/ 1377.11	**05**/ 3/ 1018.75	**05**/ 10/ 828.94	**05**/ 3/ 588.88	**05**/ 13/ 1632.34	**05**/ 4/ 1304.10
06/ 12/ 1256.36	**06**/ 3/ 913.68	**06**/ 10/ 828.94	**06**/ 3/ 588.49	**06**/ 11/ 1426.60	**06**/ 3/ 1157.35
07/ 10/ 1123.66	**07**/ 2/ 947.16	**07**/ 10/ 828.94	**07**/ 3/ 588.29	**07**/ 11/ 1232.26	**07**/ 3/ 1071.48
08/ 9/ 973.22	**08**/ 2/ 735.70	**08**/ 10/ 828.94	**08**/ 3/ 588.32	**08**/ 10/ 1144.14	**08**/ 3/ 839.10
09/ 11/ 1220.76	**09**/ 3/ 914.57	**09**/ 10/ 828.94			
10/ 10/ 1132.37	**10**/ 3/ 962.31				
11/ 10/ 1121.31	**11**/ 2/ 944.68				
12/ 9/ 1011.81					

THE RESULTS FOR 200 CUSTOMER TEST PROBLEMS BY GEHRING AND HOMBERGER (1999)					
R1..	**R2..**	**C1..**	**C2..**	**RC1..**	**RC2..**
01/ 19/ 5024.65	**01**/ 4/ 4854.36	**01**/ 20/ 2704.57	**01**/ 6/ 1931.44	**01**/ 18/ 4172.32	**01**/ 6/ 3167.78
02/ 18/ 4234.35	**02**/ 4/ 3823.58	**02**/ 18/ 3140.52	**02**/ 6/ 1863.16	**02**/ 18/ 3674.81	**02**/ 5/ 2873.11
03/ 18/ 3552.48	**03**/ 4/ 3023.78	**03**/ 18/ 2832.99	**03**/ 6/ 1808.60	**03**/ 18/ 3284.38	**03**/ 4/ 2743.38
04/ 18/ 3177.67	**04**/ 4/ 2020.95	**04**/ 18/ 2696.14	**04**/ 6/ 1754.79	**04**/ 18/ 3044.29	**04**/ 4/ 2110.85
05/ 18/ 4464.08	**05**/ 4/ 3485.74	**05**/ 20/ 2702.05	**05**/ 6/ 1879.31	**05**/ 18/ 3887.09	**05**/ 4/ 3379.67
06/ 18/ 3824.87	**06**/ 4/ 3009.36	**06**/ 20/ 2701.04	**06**/ 6/ 1857.35	**06**/ 18/ 3705.53	**06**/ 5/ 2666.24
07/ 18/ 3330.56	**07**/ 4/ 2534.06	**07**/ 20/ 2701.04	**07**/ 6/ 1850.13	**07**/ 18/ 3508.34	**07**/ 4/ 2704.06
08/ 18/ 3085.72	**08**/ 4/ 1856.32	**08**/ 19/ 2799.85	**08**/ 6/ 1822.65	**08**/ 18/ 3341.86	**08**/ 4/ 2371.97
09/ 18/ 4026.50	**09**/ 4/ 3135.40	**09**/ 18/ 2775.27	**09**/ 6/ 1848.12	**09**/ 18/ 3263.93	**09**/ 4/ 2231.63
10/ 18/ 3493.38	**10**/ 4/ 2709.33	**10**/ 18/ 2734.56	**10**/ 6/ 1808.72	**10**/ 18/ 3198.18	**10**/ 4/ 2034.94
THE RESULTS FOR 400 CUSTOMER TEST PROBLEMS BY GEHRING AND HOMBERGER (1999)					
R1..	**R2..**	**C1..**	**C2..**	**RC1..**	**RC2..**
01/ 38/ 11084.00	**01**/ 8/ 9734.34	**01**/ 40/ 7152.06	**01**/ 12/ 4116.31	**01**/ 37/ 9038.21	**01**/ 11/ 7551.22
02/ 36/ 10064.93	**02**/ 8/ 7991.14	**02**/ 37/ 7599.96	**02**/ 12/ 3986.08	**02**/ 36/ 9020.71	**02**/ 10/ 6357.11
03/ 36/ 8657.06	**03**/ 8/ 6257.51	**03**/ 36/ 7504.78	**03**/ 12/ 3923.80	**03**/ 36/ 8251.02	**03**/ 8/ 5427.41
04/ 36/ 7678.69	**04**/ 8/ 4591.97	**04**/ 36/ 7297.27	**04**/ 12/ 3837.81	**04**/ 36/ 7747.15	**04**/ 8/ 3769.80
05/ 36/ 10242.88	**05**/ 8/ 7461.57	**05**/ 40/ 7152.06	**05**/ 12/ 3945.86	**05**/ 36/ 9195.27	**05**/ 9/ 6558.59
06/ 36/ 9379.62	**06**/ 8/ 6533.67	**06**/ 40/ 7153.46	**06**/ 12/ 3885.02	**06**/ 36/ 8974.22	**06**/ 9/ 6031.59
07/ 36/ 8298.71	**07**/ 8/ 5441.64	**07**/ 40/ 7149.43	**07**/ 12/ 3935.49	**07**/ 36/ 8801.69	**07**/ 8/ 6020.05
08/ 36/ 7621.03	**08**/ 8/ 4336.30	**08**/ 38/ 7380.76	**08**/ 12/ 3845.92	**08**/ 36/ 8569.39	**08**/ 8/ 5145.53
09/ 36/ 9629.72	**09**/ 8/ 6888.98	**09**/ 37/ 7284.05	**09**/ 12/ 3959.18	**09**/ 36/ 8472.82	**09**/ 8/ 4853.95
10/ 36/ 8888.35	**10**/ 8/ 6241.62	**10**/ 36/ 7542.93	**10**/ 12/ 3791.62	**10**/ 36/ 8216.87	**10**/ 8/ 4617.50

Appendix 2

The best known solutions:

Problem: RC105; Routes: 13; Total traveled distance: 1632.34;

< 0, 98, 14, 47, 15, 16, 9, 10, 13, 17, 0 >

< 0, 39, 36, 44, 38, 40, 37, 35, 43, 0 >

< 0, 42, 61, 8, 6, 46, 4, 3, 1, 70, 0 >

< 0, 33, 76, 89, 48, 21, 25, 24, 0 >

< 0, 2, 45, 5, 7, 79, 55, 68, 0 >

< 0, 83, 19, 23, 18, 22, 49, 20, 77, 0 >

< 0, 90, 53, 66, 56, 0 >

< 0, 31, 29, 27, 30, 28, 26, 32, 34, 50, 80, 0 >

< 0, 63, 62, 67, 84, 51, 85, 91, 0 >

< 0, 72, 71, 81, 41, 54, 96, 94, 93, 0 >

< 0, 65, 82, 12, 11, 87, 59, 97, 75, 58, 0 >

< 0, 92, 95, 64, 99, 52, 86, 57, 74, 0 >

< 0, 69, 88, 78, 73, 60, 100, 0 >

Problem: RC202; Routes: 3; Total traveled distance: 1375.45;

< 0, 91, 92, 95, 85, 63, 33, 28, 26, 27, 29, 31, 30, 62, 67, 71, 72, 38, 40, 41, 81, 90, 84, 49, 20, 83, 66, 56, 50, 34, 32, 89, 24, 21, 25, 77, 75, 58, 0 >

< 0, 45, 5, 3, 1, 42, 39, 37, 36, 44, 61, 88, 73, 16, 99, 53, 78, 79, 8, 6, 46, 2, 55, 68, 43, 35, 54, 96, 93, 94, 80, 0 >

< 0, 65, 82, 98, 12, 14, 47, 15, 11, 69, 64, 19, 23, 48, 18, 76, 51, 22, 86, 87, 9, 57, 52, 10, 97, 59, 74, 13, 17, 7, 4, 60, 100, 70, 0 >

Problem: D417; Routes: 54; Total traveled distance: 3506.21;

< 0, 324, 328, 327, 323, 325, 320, 332, 351, 334, 0 >

< 0, 74, 32, 55, 50, 2, 44, 379, 368, 0 >

< 0, 407, 370, 400, 376, 406, 410, 387, 37, 310, 0 >

< 0, 304, 218, 179, 316, 315, 221, 42 0 >

< 0, 357, 359, 40, 346, 388, 52, 29, 0 >

< 0, 122, 415 ,91, 105, 417, 103, 31, 23, 0 >

< 0, 391, 366, 385, 378, 64, 142, 0 >

< 0, 215, 211, 279, 259, 254, 271, 0 >

< 0, 283, 299, 255, 276, 288, 292, 280, 183, 0 >

< 0, 75, 17, 104, 72, 35, 171, 168, 150, 0 >

< 0, 84, 12, 20, 119, 108, 16, 98, 0 >

< 0, 290, 238, 307, 264, 216, 308, 228, 0 >

< 0, 159, 158, 169, 161, 160, 146, 278, 184, 0 >

< 0, 94, 352, 348, 361, 345, 360, 355, 333, 0 >

< 0, 311, 386, 399, 382, 373, 369, 363, 412, 0 >

< 0, 416, 21, 8, 82, 48, 43, 45, 80, 0 >

< 0, 263, 241, 253, 305, 243, 318, 309, 274, 0 >

< 0, 295, 301, 256, 302, 297, 265, 186, 312, 0 >

< 0, 25, 15, 101, 86, 85, 138, 118, 0 >

< 0, 195, 217, 204, 212, 200, 197, 209, 208, 219, 207, 0 >

< 0, 178, 188, 266, 272, 177, 296, 201, 230, 176, 0 >

< 0, 285, 205, 173, 232, 185, 222, 199, 231, 175, 0 >

< 0, 344, 336, 337, 339, 340, 338, 111, 0 >

< 0, 54, 4, 93, 141, 83, 5, 57, 90, 0 >

< 0, 78, 65, 117, 19, 73, 24, 9, 130, 0 >

< 0, 165, 155, 166, 147, 133, 0 >

< 0, 408, 404, 397, 392, 398, 405, 384, 390, 0 >

< 0, 353, 113, 33, 6, 14, 81, 100, 0 >

< 0, 47, 49, 46, 1, 143, 109, 28, 0 >

< 0, 395, 403, 409, 394, 145, 347, 343, 393, 396, 413, 0 >

< 0, 245, 258, 275, 257, 247, 246, 291, 198, 0 >

< 0, 313, 252, 367, 411, 381, 287, 260, 220, 0 >

< 0, 372, 18, 124, 116, 30, 13, 88, 99, 0 >

< 0, 242, 250, 306, 244, 251, 317, 262, 0 >

< 0, 213, 190, 289, 237, 236, 202, 210, 206, 267, 0 >

< 0, 180, 181, 191, 273, 239, 277, 240, 286, 0 >

< 0, 269, 174, 223, 261, 284, 194, 293, 229, 0 >

< 0, 226, 187, 214, 248, 235, 225, 196, 189, 224 0 >

< 0, 102, 354, 356, 362, 319, 321, 329, 326, 342, 0 >

< 0, 76, 7, 114, 69, 112, 70, 68, 95, 0 >

< 0, 144, 303, 87, 61, 60, 34, 136, 0 >

< 0, 314, 203, 268, 110, 71, 121, 77, 26, 0 >

< 0, 123, 156, 157, 163, 162, 167, 149, 152, 0 >

< 0, 374, 377, 364, 401, 249, 371, 375, 383, 414, 39, 0 >

< 0, 341, 335, 322, 330, 358, 89, 0 >

< 0, 298, 389, 365, 402, 137, 380, 38, 0 >

< 0, 125, 115, 63, 62, 126, 79, 0 >

< 0, 139, 66, 41, 170, 59, 58, 135, 0 >

< 0, 22, 27, 3, 127, 128, 51, 294, 0 >

< 0, 164, 153, 172, 151, 154, 148, 36, 106, 0 >

< 0, 282, 11, 120, 140, 131, 53, 92, 0 >

< 0, 270, 300, 227, 193, 192, 233, 234, 0 >

< 0, 96, 350, 349, 331, 97, 134, 0 >

< 0, 281, 182, 67, 10, 129, 56, 107, 132, 0 >

Problem: E417; Routes: 54; Total traveled distance: 3801.64;

< 0, 395, 409, 397, 404, 145, 394, 403, 396, 413, 393, 0 >	< 0, 226, 192, 193, 217, 220, 206, 196, 293, 267, 0 >
< 0, 324, 328, 323, 325, 327, 320, 326, 342, 361, 0 >	< 0, 283, 386, 402, 109, 18, 183, 310, 0 >
< 0, 180, 191, 273, 239, 243, 305, 256, 186, 0 >	< 0, 74, 2, 50, 55, 113, 415, 143, 28, 0 >
< 0, 22, 45, 31, 91, 105, 64, 13, 0 >	< 0, 341, 356, 330, 329, 348, 332, 347, 343, 0 >
< 0, 104, 17, 114, 95, 72, 89, 155, 0 >	< 0, 25, 15, 101, 108, 86, 99, 136, 0 >
< 0, 182, 227, 235, 300, 214, 187, 210, 202, 237, 0 >	< 0, 159, 163, 162, 157, 158, 169, 160, 278, 41, 0 >
< 0, 281, 212, 307, 264, 260, 313, 288, 248, 0 >	< 0, 178, 177, 296, 170, 152, 146, 184, 201, 0 >
< 0, 269, 189, 311, 259, 301, 242, 277, 224, 0 >	< 0, 270, 37, 252, 39, 367, 364, 287, 255, 0 >
< 0, 377, 374, 411, 369, 412, 363, 370, 203, 0 >	< 0, 304, 261, 291, 232, 284, 173, 205, 218, 0 >
< 0, 314, 381, 299, 207, 289, 219, 190, 194, 0 >	< 0, 245, 258, 181, 266, 188, 240, 286, 0 >
< 0, 195, 225, 276, 389, 137, 380, 38, 280, 0 >	< 0, 416, 32, 8, 20, 81, 14, 6, 119, 0 >
< 0, 26, 88, 100, 16, 12, 121, 71, 0 >	< 0, 295, 211, 279, 309, 302, 297, 265, 0 >
< 0, 213, 204, 236, 274, 271, 254, 198, 231, 0 >	< 0, 76, 111, 337, 70, 68, 102, 75, 0 >
< 0, 110, 127, 51, 77, 129, 79, 131, 0 >	< 0, 164, 147, 36, 172, 171, 168, 154, 0 >
< 0, 47, 49, 46, 82, 1, 48, 43, 0 >	< 0, 357, 346, 359, 40, 345, 360, 349, 331, 0 >
< 0, 353, 350, 97, 134, 96, 85, 0 >	< 0, 391, 366, 385, 379, 23, 103, 417, 0 >
< 0, 238, 290, 200, 197, 208, 209, 216, 222, 185, 0 >	< 0, 372, 249, 383, 401, 371, 414, 375, 233, 234, 0 >
< 0, 67, 60, 61, 53, 135, 58, 140, 0 >	< 0, 263, 306, 244, 251, 317, 318, 272, 0 >
< 0, 388, 405, 408, 384, 398, 392, 390, 44, 0 >	< 0, 298, 400, 365, 368, 378, 116, 142, 0 >
< 0, 215, 253, 241, 250, 262, 312, 199, 229, 0 >	< 0, 144, 84, 98, 118, 90, 138, 34, 0 >
< 0, 407, 382, 399, 373, 406, 376, 410, 387, 308, 0 >	< 0, 165, 151, 148, 35, 150, 166, 153, 0 >
< 0, 315, 179, 316, 221, 120, 11, 132, 0 >	< 0, 352, 321, 319, 322, 358, 344, 7, 0 >
< 0, 69, 336, 339, 354, 334, 333, 355, 83, 0 >	< 0, 122, 80, 27, 30, 124, 268, 292, 228, 0 >
< 0, 42, 56, 130, 106, 65, 78, 66, 0 >	< 0, 285, 223, 174, 247, 246, 257, 275, 176, 230, 0 >
< 0, 4, 93, 5, 57, 52, 29, 33, 21, 0 >	< 0, 294, 3, 128, 10, 92, 107, 175, 0 >
< 0, 123, 161, 156, 133, 73, 24, 9, 115, 0 >	< 0, 54, 87, 63, 62, 126, 125, 0 >
< 0, 282, 139, 59, 117, 19, 149, 167, 303, 0 >	< 0, 141, 94, 351, 335, 362, 340, 338, 112, 0 >

Acknowledgements

This work was partially supported by the Emil Aaltonen Foundation and TOP project funded by Research Council of Norway. This support is gratefully acknowledged. We would also like to thank three anonymous referees and associate editor for their valuable comments.

References

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.

Bachem, A., W. Hochstättler, M. Malich. 1996. The simulated trading heuristic for solving vehicle routing problems. *Discrete Applied Mathematics* **65** 47–72.

Badeau, P., M. Gendreau, F. Guertin, J.-Y. Potvin, E. Taillard. 1997. A parallel tabu search heuristic for the vehicle routing problem with time windows. *Transportation Research – C* **5** 109–122.

Baker, E.K., J.R. Schaffer. 1986. Solution improvement heuristics for the vehicle routing and scheduling problem with time window constraints. *American Journal of Mathematical and Management Sciences* **6** 261–300.

Barnes, J.W., W.B. Carlton. 1995. A tabu search approach to the vehicle routing problem with time windows. Presented at the Fall INFORMS Conference, New Orleans, LA.

Battiti, R., G. Tecchiolli. 1994. The reactive tabu search. *ORSA Journal on Computing* **6**, 126–140.

Berger, J., M. Salois, R. Begin. 1998. A hybrid genetic algorithm for the vehicle routing problem with time windows. *Lecture Notes in Artificial Intelligence* **1418**, AI'98, Advances in Artificial Intelligence, Vancouver, BC, Canada, 114–127.

Blanton, J.L., R.L. Wainwright. 1993. Multiple vehicle routing with time and capacity constraints using genetic algorithms. S. Forrest, ed. *Proceedings of the Fifth International Conference on Genetic Algorithms*. Morgan Kaufmann Publishing, San Francisco. 452–459.

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

Brandão, J. 1999. Metaheuristic for the vehicle routing problem with time windows. S. Voss, S. Martello, I.H. Osman, C. Roucairol, eds. *Meta-heuristics – Advances and Trends in Local Search Paradigms for Optimization*. Kluwer Academic Publishers, Boston. 19–36.

Bräysy, O. 1999a. A hybrid genetic algorithm for the vehicle routing problem with time windows. Licentiate thesis, University of Vaasa, Vaasa, Finland.

Bräysy, O. 1999b. A new algorithm for the vehicle routing problem with time windows based on the hybridization of a genetic algorithm and route construction heuristics. *Proceedings of the University of Vaasa*, *Research papers* **227**.

Bräysy, O., J. Berger, M. Barkaoui. 2000. A new hybrid evolutionary algorithm for the vehicle routing problem with time windows. Presented at the Route 2000-Workshop, Skodsborg, Denmark.

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

Carlton, W.B. 1995. A tabu search approach to the general vehicle routing problem. Ph.D. thesis, University of Texas, Austin, U.S.A.

Caseau, Y., F. Laburthe. 1999. Heuristics for large constrained vehicle routing problems. *Journal of Heuristics* **5** 281−303.

Chiang, W.C., R.A. Russell. 1996. Simulated annealing metaheuristics for the vehicle routing problem with time windows. *Annals of Operations Research* **63** 3−27.

Chiang, W.C., R.A. Russell. 1997. A reactive tabu search metaheuristic for the vehicle routing problem with time windows. *INFORMS Journal on Computing* **9** 417−430.

Clarke, G., J.W. Wright. 1964. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research* **12** 568−581.

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

Cordeau, J.-F., G. Laporte, A. Mercier. 2001. A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational Research Society* **52**, 928–936.

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

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

De Backer, B., V. Furnon, P. Kilby, P. Prosser, P. Shaw. 2000. Solving vehicle routing problems using constraint programming and metaheuristics. *Journal of Heuristics* **6** 501−523.

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

Desrosiers, J., Y. Dumas, M.M. Solomon, F. Soumis. 1995. Time constrained routing and scheduling. M. Ball, ed. *Handbooks in Operations Research and Management Science 8*: *Network Routing*. Elsevier Science Publishers, Amsterdam. 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, U.S.A.

Gambardella, L.M., E. Taillard, G. Agazzi. 1999. MACS-VRPTW: a multiple ant colony system for vehicle routing problems with time windows. D. Corne, M. Dorigo and F. Glover, eds. *New Ideas in Optimization*. McGraw-Hill, London. 63−76.

Garcia, B.-L., J.-Y. Potvin, J.-M. Rousseau. 1994. A parallel implementation of the tabu search heuristic for vehicle routing problems with time window constraints. *Computers & Operations Research* **21** 1025−1033.

Gehring, H., J. Homberger. 1999. A parallel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows. K. Miettinen, M. Mäkelä and J. Toivanen, eds. *Proceedings of EUROGEN99 - Short Course on Evolutionary Algorithms in Engineering and Computer Science*, *Reports of the Department of Mathematical Information Technology*, Series A. Collections, No. A 2/1999. University of Jyväskylä, Jyväskylä. 57−64.

Gehring, H. and J. Homberger. 2001. Parallelization of a two-phase metaheuristic for routing problems with time windows. *Asia-Pacific Journal of Operational Research* **18** 35−47.

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

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

Golden, B.L., A.A. Assad. 1986. Perspectives on vehicle routing: exciting new developments. *Operations Research* **34** 803−809.

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

Hansen, P., N. Mladenovic. 2000. Variable neighborhood search: A Chapter of Handbook of Applied Optimization. Working paper, Group for Research in Decision Analysis, University of Montreal, Canada.

Homberger, J., H. Gehring. 1999. Two evolutionary meta-heuristics for the vehicle routing problem with time windows. *INFOR* **37** 297–318.

Kilby, P., P. Prosser, P. Shaw. 1999. Guided local search for the vehicle routing problem with time windows. S. Voss, S. Martello, I.H. Osman and C. Roucairol, eds. *META-HEURISTICS Advances and Trends in Local Search Paradigms for Optimization*. Kluwer Academic Publishers, Boston. 473–486.

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 Science* **33** 101–116.

Kontoravdis, G.A., J.F. Bard. 1995. A GRASP for the vehicle routing problem with time windows. *Journal on Computing* **7** 10–23.

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.

Liu, F.-H., S.-Y. Shen. 1999. A route-neighborhood-based metaheuristic for vehicle routing problem with time windows. *European Journal of Operations Research* **118** 485–504.

Mladenovic, N., P. Hansen. 1997. Variable neighborhood search. *Computers & Operations Research* **24** 1097–1100.

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

Osman, I.H. 1993. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problems. *Annals of Operations Research* **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. *European Journal of Operational Research* **66** 331–340.

Potvin, J.-Y., J.-M. Rousseau. 1995. An exchange heuristic for routeing problems with time windows. *Journal of the Operational Research Society* **46** 1433–1446.

Potvin, J.-Y., C. Robillard. 1995. Clustering for vehicle routing with a competitive neural network. *Neurocomputing* **8** 125–139.

Potvin, J.-Y., S. Bengio. 1996. The vehicle routing problem with time windows part II: genetic search. *Journal on Computing* **8** 165–172.

Potvin, J.-Y., T. Kervahut, B.L. Garcia, J.-M. Rousseau. 1996a. The vehicle routing problem with time windows part I: tabu search. *Journal on Computing* **8** 157−164.

Potvin, J.-Y., D. Dube, C. Robillard. 1996b. A hybrid approach to vehicle routing using neural networks and genetic algorithms. *Applied Intelligence* **6** 241-252.

Rochat, Y., E. Taillard. 1995. Probabilistic diversification and intensification in local search for vehicle routing. *Journal of Heuristics* **1** 147−167.

Rousseau, L.-M., M. Gendreau, G. Pesant. 2000. Using constraint-based operators to solve the vehicle routing problem with time windows. Working Paper, Centre for Research on Transportation, University of Montreal, Montreal, Canada. To appear in Journal of Heuristics.

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

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

Savelsbergh, M.W.P. 1992. The vehicle routing problem with time windows: minimizing route duration. *Journal on Computing* **4**, 146−154.

Schulze, J., T. Fahle. 1999. A parallel algorithm for the vehicle routing problem with time window constraints. *Annals of Operations Research* **86** 585−607.

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 and 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. 1987. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research* **35** 254−265.

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 and A. Assad, eds. *Vehicle Routing: Methods and Studies*. Elsevier Science Publishers, Amsterdam 85−106.

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

Tan, K. C., L. H. Lee, K. Q. Zhu. 2000. Heuristic methods for vehicle routing problem with time windows. *Proceedings of the 6th International Symposium on Artificial Intelligence & Mathematics*, Ft. Lauderdale, Florida.

Tan, K. C., L. H. Lee, K. Ou. 2001. Hybrid genetic algorithms in solving vehicle routing problems with time window constraints. *Asia-Pacific Journal of Operational Research* **18**, 121−130.

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

Thangiah, S., I. Osman, T. Sun. 1994. Hybrid genetic algorithm, simulated annealing and tabu search methods for vehicle routing problems with time windows. Working Paper UKC/IMS/OR94/4, Institute of Mathematics and Statistics, University of Kent, Canterbury.

Thangiah, S. 1995. Vehicle routing with time windows using genetic algorithms. L. Chambers, ed. *Application Handbook of Genetic Algorithms*: *New Frontiers*, *Volume II.* CRC Press, Boca Raton. 253−277.

Thangiah, S.R., I.H. Osman, R. Vinayagamoorthy, T. Sun. 1995. Algorithms for the vehicle routing problems with time deadlines. *American Journal of Mathematical and Management Sciences* **13** 323−355.

Thompson, P. M., H.N. Psaraftis. 1993. Cyclic transfer algorithms for multivehicle routing and scheduling problems. *Operations Research* **41** 935−946.

Van Landeghem, H.R.G. 1988. A bi-criteria heuristic for the vehicle routing problem with time windows. *European Journal of Operations Research* **36** 217−226.

Voudouris, C. 1997. Guided local search for combinatorial problems. Ph.D. thesis, Department of Computer Science, University of Essex, Colchester, UK.

Voudouris, C., E. Tsang. 1998. Guided local search. *European Journal of Operations Research* **113** 80−119.