Available online at www.sciencedirect.com
SCIENCE DIRECT®
European Journal of Operational Research 159 (2004) 586–605
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/dsw

Discrete Optimization

A multi-start local search algorithm for the vehicle routing problem with time windows

Olli Bräysy [a,*], Geir Hasle [a], Wout Dullaert [b]

[a] SINTEF Applied Mathematics, Department of Optimisation, P.O. Box 124 Blindern, N-0314 Oslo, Norway
[b] Ufsia-Ruca Faculty of Applied Economics, University of Antwerp, Prinsstraat 13, 2000 Antwerp, Belgium

Received 25 September 2002; accepted 11 December 2002
Available online 8 October 2003

Abstract

In this paper a multi-start local search (MSLS) heuristic is proposed for the vehicle routing problem with time windows (VRPTW). In VRPTW the objective is to design least cost routes for a fleet of identical capacitated vehicles to service geographically scattered customers within pre-specified service time windows. The suggested approach is based on a MSLS framework and several new improvement heuristics. A new speedup technique is introduced for the construction heuristics, and the results of the MSLS are post-optimized by a threshold accepting post-processor. Experimental results on 358 benchmark problems from the literature show that the suggested method is highly efficient and competitive.

Keywords: Routing; Time windows; Metaheuristics

1. Introduction

In the vehicle routing problem with time windows (VRPTW) the objective is to design least cost routes from a single location (the depot) to a set of geographically scattered points (the customers). The routes must be designed in such a way that each point is visited only once by exactly one vehicle within a given time interval. All routes start and end at the depot, and the total demand of all points on one particular route must not exceed the capacity of the vehicle.

Triggered by a vast array of applications and its intrinsic difficulty, the VRPTW has been the subject of intensive research. As solving the NP-hard VRPTW (Lenstra and Rinnooy Kan, 1981) to optimality remains hard, even for problem instances involving just one hundred customers (Kohl, 1995; Larsen, 1999; Kohl et al., 1999; Cook and Rich, 1999), research has focused on heuristic approaches. These VRPTW heuristics can be divided in construction heuristics, improvement heuristics (local searches) and metaheuristics. Construction heuristics build routes sequentially or in parallel. Sequential route construction heuristics build routes one at a time until the route's scarce resources (e.g. capacity, maximum driving distance or time) are depleted, without violating time window constraints. Parallel route

* Corresponding author.
E-mail addresses: olli.braysy@sintef.no (O. Bräysy), geir.hasle@sintef.no (G. Hasle), wout.dullaert@ua.ac.be (W. Dullaert).

0377-2217/$ - see front matter © 2003 Elsevier B.V. All rights reserved.
doi:10.1016/S0377-2217(03)00435-1

construction heuristics create several routes simultaneously. Different variants of construction heuristics can be found in Solomon (1987), Potvin and Rousseau (1993), Bramel and Simchi-Levi (1996), Dullaert and Bräysy (in press) and Ioannou et al. (2001).

Improvement heuristics are used to obtain higher quality solutions by trying modifications to the incumbent solution. For the most successful applications to the VRPTW, see Thompson and Psaraftis (1993), Potvin and Rousseau (1995), Russell (1995), Shaw (1997), Shaw (1998), Caseau et al. (1999) and Cordone and Wolfler-Calvo (2001). Construction and improvement heuristics are discussed in detail in Bräysy and Gendreau (in press a). Metaheuristics are used to guide construction and improvement heuristics to escape local optima. High-quality results are obtained in Rochat and Taillard (1995), Taillard et al. (1997), Gambardella et al. (1999), Homberger and Gehring (1999), Liu and Shen (1999), Bent and Van Hentenryck (in press), Berger et al. (2003), Cordeau et al. (2001), Gehring and Homberger (2001), Ibaraki et al. (in press), Li et al. (2003), Czech and Czarnas (2002), and Rousseau et al. (2002). For more details, we refer to the recent survey on metaheuristics by Bräysy and Gendreau (in press a).

Recent literature reviews (Bräysy and Gendreau, in press a,b) reveal that the previously developed heuristics and metaheuristics for the VRPTW show significant variability in performance. The solutions obtained with the traditional construction and improvement heuristics are often far away from optimum, and these methods are also highly dependent on the initial solutions. Better results can be obtained with different metaheuristics, but they often require considerable computational effort, do not scale well, and fail to convincingly provide a single robust and successful technique. Given the practical importance of the VRPTW, further research on this topic is justified.

The main contribution of this paper is the development of an efficient and robust multi-start local search (MSLS) heuristic for the VRPTW. We consider the higher efficiency of our approach compared to previous approaches practically relevant, especially in dynamic real-life problems. The suggested method uses a two-phase approach. In the first phase, a fast construction heuristic is used to generate several initial solutions. Then, injection trees (IT), an extension of the well-known ejection chain (EC) approach (Glover, 1991, 1992), are used to reduce the number of routes. In the second phase, two new improvement heuristics, based on CROSS-exchanges (Taillard et al., 1997) are applied for distance minimization. Together, Phase 1 and Phase 2 constitute the MSLS algorithm. The best solution identified by the MSLS is post-optimized using a threshold accepting (TA) post-processor, introduced in Bräysy et al. (in press). The TA post-processor is powered by the IOPT intra-route improvement heuristic (Bräysy, 2001) and a new and highly efficient inter-route improvement heuristic (GENICROSS), based on the work of Gendreau et al. (1992) and Taillard et al. (1997). The results on a set of 358 benchmark problems from the studies of Solomon (1987), Gehring and Homberger (1999) and Russell (1995) illustrate the power of the suggested local search to quickly generate high quality solutions and confirm the merits of the TA post-processor. The resulting hybrid method is shown to be fast, cost-effective and highly competitive.

The remainder of this paper is arranged as follows. First, an overview of the solution approach is given. Then, the hybrid sequential insertion heuristic as well as the new improvement heuristics for minimizing the number of routes and travelled distance are described in Section 2. Section 3 presents a description of the TA post-processor. The empirical investigation is described in Section 4. Conclusions are drawn in Section 5.

2. A new multi-start local search heuristic

The suggested local search consists of two phases with an optional post-optimization procedure (see Algorithm 1). In the first phase, a new sequential insertion heuristic based on Bräysy (2001) is used to quickly generate a set of initial solutions. IT, a new approach related to the concept of EC (Glover, 1991; Glover, 1992) are applied to reduce the number of routes. In Phase 2, modified intra- and inter-route CROSS-exchanges

Algorithm 1. Two-phase MSLS for the VRPTW

PHASE 1: BUILDING SOLUTIONS WITH THE SMALLEST NUMBER OF ROUTES (SEE SECTION 2.1)

- Read the problem data from file. Set $\alpha_1 = 0.7$

while $\alpha_1 \leqslant 1$ **do**
- Set $\alpha_2 = 1 - \alpha_1$ and $\alpha_3 = 0.5$

while $\alpha_3 \leqslant 1.7$ **do**
for 12 iterations **do**
- Use the construction heuristic to create an initial solution S_i.
- Apply the Injection Tree procedure on S_i until no more routes can be eliminated.
- If the number of routes r_i in S_i is less than or equal to the smallest number of routes found so far, r_b, set $r_b = r_i$ and store S_i.

end for
- Set $\alpha_3 = \alpha_3 + 0.2$

end while
- Set $\alpha_1 = \alpha_1 + 0.1$

end while

PHASE 2: DISTANCE IMPROVEMENT ON THE SOLUTIONS FROM PHASE 1 (SEE SECTION 2.2)

for all stored solutions S_i, for which r_i is equal to r_b **do**
while no more improvements can be found in S_i **do**
for all geographically close pairs of routes in S_i **do**
- Try to improve the current pair of routes in S_i using modified CROSS exchanges.

end for
for all individual routes in S_i **do**
- Try to improve the current route using modified CROSS exchanges.

end for
end while
- If S_i is better than the best solution found so far, S_b, set $S_b = S_i$.

end for

POST-OPTIMIZATION: DISTANCE IMPROVEMENT ON THE BEST SOLUTION FROM PHASE 2 (SEE SECTION 3)

- Try to improve S_b with the Threshold Accepting post-optimization.

(Taillard et al., 1997) are invoked to reduce the total distance of the solutions with the minimum number of routes. The best solution obtained by the MSLS after phases 1 and 2 can be post-optimized by a TA post-processor to further reduce the total distance of the solution.

2.1. Phase 1: Initial solutions and route reduction

2.1.1. Building initial solutions

Because of the high dependency between the initial solutions and the final output, most metaheuristics work on a set of initial solutions. The same strategy is used also here. For generating the initial solutions we use a modification of the sequential cheapest insertion heuristic of Bräysy (2001). We initialize routes by selecting randomly an, as of yet, unrouted customer among the customers farthest from the depot or among the customers having earliest deadline for service. The initialization criterion is selected randomly before starting construction of a new solution. After adding the initialization customer to the route, unrouted customers are evaluated for insertion. To speed up the construction heuristic, only customers that are geographically close to at least one of the previously inserted customers on the route are considered for insertion. We define an unrouted customer to be geographically close if its distance to one of the customers on the current partial

route is less than $d = 0.30$ times the maximum distance between any pair of customers in the problem. A similar strategy of limiting the search according to the closeness of customers has been used before for instance in Shaw (1997), Shaw (1998), Liu and Shen (1999) and Berger et al. (2003). The insertion cost of an unrouted customer is defined as the weighted combination of additional detour and waiting time needed to insert a customer at its best feasible insertion position in the 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. Therefore, the selection of these distant customers is favored by subtracting from the insertion cost the distance of the corresponding customer to the depot multiplied by a user defined parameter α_3. Once all insertion costs and insertion places are evaluated, the cheapest customer is chosen to be inserted in the route. More formally the cost function C_u for customer u is

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

where

$$D_u = d_{iu} + d_{uj} - d_{ij}, \tag{2}$$

$$W_u = W_u^a - W_u^b, \tag{3}$$

$$\alpha_1 + \alpha_2 = 1,\ \alpha_3 > 0. \tag{4}$$

Notations d_{iu}, d_{uj} and d_{ij} refer to the distance between the corresponding pair of customers (i,u), (u,j) and (i,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 depot to customer u, and α_1, α_2 and α_3 are parameter values determined by the user.

We used the following values during all computational experiments: α_1: 0.6–1.0 (in increments of 0.1 units) and α_3: 0.5–1.7 (in increments of 0.2 units). [1] All possible combinations of α_1 and α_3 values are tried 12 times because of the randomly selected seed customers. This is because the initialization scheme seems to have high impact on the results (Bräysy, 2001). Therefore, 420 different initial solutions are created in total.

To speed up checking the time window constraints on each insertion, we used push-forward and push-backward strategies, introduced in Solomon et al. (1988), and keep arrival times and latest possible arrival times at each customer in memory. In addition, a new speedup technique was created to speed up the evaluation of the cost of each move. As defined above, the insertion cost is a weighted combination of distance and waiting time (see Eq. (1)). The additional detour can be easily calculated in constant time using Eq. (2). For evaluating the waiting time or the duration of the route, the common procedure is to loop through the route once to evaluate each insertion position within the route and sum up the waiting times at all customers. However, by controlling the order in which the insertion places within the route are considered, it is possible to evaluate the change in the total waiting time (and duration) in the route in constant time using Eq. (3).

The key idea is to consider the insertion places within the route in the order in which the customers are currently serviced by the vehicle. That is, we evaluate first the insertion between the depot and the first customer, then between the first customer and the second customer on the current partial route, and so on. Each time, when evaluating an insertion position, we accumulate the waiting time at the customers considered so far. More precisely, when evaluating the insertion between the depot and the first customer we calculate the waiting time at the first customer before insertion. Similarly, when considering insertion between the first and second customer we calculate the waiting time before insertion at the second customer and so on. Each time we accumulate the waiting time such that when we consider insertion between customers i and j, we know the sum of waiting times at customers $1,\ldots,j$. Let us denote this accumulated waiting time by w_a. In addition, we keep in memory the total waiting time of the route before insertion, w_t. We also calculate the waiting time at the newly inserted customer, w_u, and the change in arrival time at j, p_f, which is often called "push forward" (Solomon, 1987). Three different cases can be distinguished. When

[1] As the value of α_2 is determined by α_1, we need to specify the value only for α_1 and α_3.

evaluating insertion between the depot and the first customer the total waiting time after the insertion, $\tilde{w}_t$, is $\tilde{w}_t = w_u$ if $p_f > w_t$, and $\tilde{w}_t = w_t + w_u - p_f$ otherwise. Both results are based on the fact that the push forward generated by inserting a customer between the depot and the first customer can be eliminated by waiting time stored in the route at the first customer and its successors. Similarly, when considering the insertion between the last customer and the depot, $\tilde{w}_t = w_t + w_u$. In other cases $\tilde{w}_t = w_u + w_a$ if $p_f > w_t - w_a$, and $\tilde{w}_t = w_t + w_u - p_f$ otherwise. Thus, the total waiting time of the route and also duration of the route can be calculated in constant time. As a result, the insertion heuristic is capable of generating about 5000 feasible solutions for Solomon's (1987) 100 customer problem instances in 1 second on a 700 MHz PC.

2.1.2. Reducing the number of routes

After creating an initial solution, an attempt is made to reduce the number of routes in the solution using a new heuristic operator, named IT. The IT are applied to all created initial solutions and the procedure is repeated as long as one can reduce the number of routes. The idea of using separate strategies for reducing the number of routes in a VRPTW is not new. It has been used for example in Gambardella et al. (1999), Homberger and Gehring (1999), Gehring and Homberger (1999), Gehring and Homberger (2001), Berger et al. (2003), Bent and Van Hentenryck (in press), and Liu and Shen (1999). However, only Gehring and Homberger (1999), Gehring and Homberger (2001), and Bent and Van Hentenryck (in press) have a separate phase for reducing the number of routes in the beginning of the procedure, and none of them uses a special operator for that purpose as is done in our approach.

IT share similarities with EC. EC (Glover, 1991; Glover, 1992) have been applied often in recent VRP methodologies (Rousseau et al., 2002; Bräysy, 2001; Rego, 1998, 2001; Caseau et al., 1999). In EC the basic idea is to combine series of simple moves into a compound move. In a VRP context these simple moves refer to removal of a customer from its route and re-insertion of the removed customer in another route. The goal is to "make room" for a new customer in a route by first removing another customer from the same route. In each phase within the EC, one customer remains unrouted. The removal and insertion procedures are repeated until one can insert a customer to another route without the need to remove (eject) any customer. Here we present IT as an extension to the EC and apply it as a route reduction procedure. Our IT was implemented on the EC procedure of Bräysy (2001).

Instead of first ejecting a customer from a route to make room for a new customer, IT directly inserts customers in the target route, even if it leads to violation of time window or capacity constraints. The rationale behind this is that by inserting customer u directly in another route r_t, a lot of computational effort can be saved compared to repeating the insertion of u into r_t each time a customer is removed from r_t. Moreover, contrary to EC that remove one customer at a time from the target route r_t, IT conditionally remove an unlimited number of customers from r_t. More than one customer can be removed only if they can be inserted directly in a neighboring route $r_n \neq r_t$, $r_n \neq r_e$, without having to remove customers there to maintain feasibility. If these re-insertions are not feasible, the successful insertions are reversed, and we proceed as in standard breadth-first search EC. That is, we consider all possible removals of one customer from r_t that make it feasible after inserting customer u. Here we start from the customers causing highest increase in route distance, and consider first removing and re-inserting them in alternate locations. This way we consider first all possible chains involving two insertions and one ejection, then chains involving three insertions and two ejections and so on.

The advantage of allowing multiple ejections in IT compared to EC is that it makes the procedure less dependent on the type of customers. That is, EC are dependent on the fact that the customer removed from the target route, and the customer inserted in the target route, u, have similar demand and time windows etc. For example, if the demand of u is much larger than the average demand, the EC approach often fails. Similarly, it is often difficult to relocate only the customers on r_t having similarities with u to alternate routes. The IT

approach can therefore be considered to be more flexible than EC.

In the implementation of the IT a number of speed-up techniques were used to reconcile the IT's power with computation time considerations. Since it is computationally easier to eliminate shorter routes, they are considered first, and in case of large-scale problems only a limited set of shortest routes are considered for elimination. Only geographically close routes are considered for insertion, distant routes are disregarded. More precisely, if the distance between the customer u currently considered for insertion and all the customers on the given route r_t exceeds $\bar{d} = 0.30$ times the maximum distance between any pair of customers in the problem, we do not even try to insert u into r_t. Good values for $\bar{d}$ are problem dependent as they depend on the problem structure, problem size, etc. One could also determine the value based on the current search status (number of chains in the memory). A limited sensitivity analysis revealed that $\bar{d} = 0.30$ gave on average the best results. Memory requirements of the breadth-first approach are kept reasonable by imposing a maximum on the number of trees stored into memory (1000) and their depth $\bar{c} = 8$.

The idea of calculating accumulated waiting time after each feasible insertion position is also used to evaluate the changes in objective function quickly. To further speed-up the procedure, insertions that increase the distance of the target route more than a dynamically adjusted limit $\bar{l}$ are ignored for feasibility checking. The rationale is that checking feasibility of a move takes significantly more time than evaluating the cost of the move in VRPTW context (Bräysy, 2001). The starting value of $\bar{l}$ is $\bar{l} = 1.25$, and it is dynamically adjusted in steps of 0.1 units to control the complexity of the search such that only the most "promising" trees are kept in memory.

As for the construction heuristic, push-forward and push-backward strategies (Solomon et al., 1988) are applied, and information on arrival times and latest possible arrival times at each customer are maintained in memory to speed-up the feasibility checks. As in the EC approach of Bräysy (2001), an efficient reordering procedure is embedded in the IT procedure. The goal is to increase the number of feasible relocations by trying to reduce lateness in the target route using simple intra-route re-insertions with first-accept strategy. To control the complexity of the search, the number of these re-insertions is limited to $\bar{i} = 5$.

2.2. Phase 2: Distance improvement

The distance improvement is performed with modifications of the well-known CROSS-exchanges of Taillard et al. (1997), and it is applied only to solutions having the minimum number of routes. CROSS-exchanges work by relocating or exchanging a segment of consecutive customers between two routes while preserving orientation. Three modifications are introduced to widen the search. First, we consider inserting customers in the current segment also in inverted order in the alternative route. This was found to provide valuable improvements for VRPTWs with loose service time windows. Second, the neighborhood of the original CROSS exchanges is extended. In Taillard et al. (1997) the segment from Route 2 is always inserted to a location, where the segment from Route 1 was removed. In our implementation we consider in addition the most promising insertion position (before the geographically closest customer) for each segment removed from Route 2. Third, the most promising moves are tried first and are implemented in a first-accept strategy. These moves consist of first removing segments that include customers closest to the customers on the other route and inserting them selectively. The selected segments are first considered for insertion between customers that are closest to it on the other route.

A number of implementation choices are needed to reduce computation time as much as possible. As checking feasibility of a route in a VRPTW takes significantly more time than evaluating impact of the move on the objective function Bräysy (2001), computation time is reduced by evaluating the cost of the exchange first and checking feasibility only if an improvement is found. In checking the feasibility of the time window constraints, we use the push-forward, and push-backward techniques of Solomon et al. (1988) and maintain in memory the arrival times and latest possible arrival times at each customer.

In case some infeasibilities are found, further checks are disregarded. The maximum segment length was set to 5.

3. Post-optimization

In this Section we describe a post-optimization technique that uses two improvement operators guided by the TA metaheuristic to further reduce distance in the best solutions found in Phase 2.

TA (Dueck and Scheurer, 1990; Dueck et al., 1993) is a modification of the well-known simulated annealing (SA) metaheuristic (Metropolis et al., 1953; Kirkpatrick et al., 1983; Aarts et al., 1997). SA can be used to guide any improvement heuristic in the following way. It always accepts moves to neighboring solutions that do not worsen the objective function value. More precisely, the solution S in the neighborhood $N(S)$ is accepted as the new current solution if $\Delta \leqslant 0$, where $\Delta = C(S') - C(S)$ in which C denotes the objective function. To allow the search to escape a local optimum, a probabilistic approach is used to direct the search. A move that worsens the objective function value is accepted with a probability $e^{-\Delta/T}$ if $\Delta > 0$, where T is a parameter called the "temperature". The value of T is usually varied from a relatively large value to a small value close to zero according to a "cooling schedule", which specifies the initial, and temperature values at each stage of the algorithm.

Dueck and Scheurer (1990) and Dueck et al. (1993) simplified the SA procedure by leaving out the stochastic element in accepting worse solutions. Instead they introduced a deterministic threshold, t, and suggest accepting a worse solution if its difference to the incumbent solution is smaller or equal to the threshold. The new procedure was named TA and it was indicated to perform better than SA. The key components of TA are the function $g(t)$ that determines the lowering of the threshold during the course of the procedure, stopping criteria as well as the methods used to create initial and neighboring solutions.

Algorithm 2 describes the implementation of our TA post-processor for the VRPTW. The search consists of a user-defined number of iterations ($\bar{n} = 500$). Each iteration starts by putting the routes in the current solution in random order to further diversify the search. Accepting worse solutions in the inter- and intra-route local search operators is controlled by the current value of the threshold t. A modification of the current solution by the local search operators consists of replacing a set of arcs by a new set of arcs. It is accepted if the ratio of the new arc lengths to the old arc lengths is smaller than $(1 + t)$. More precisely, this ratio is calculated by summing up the new arcs lengths and dividing that sum with the total length of the old arcs.

The level of the threshold is determined by the linear function $g(t)$. Starting from a user-defined maximum threshold $t_{\max} = 1$, the threshold

Algorithm 2. Threshold Accepting post-processor for the VRPTW

(1) Read a feasible solution, S_i, created by the MSLS (see Section 2). Set $t = t_{\max}$ and $S_b = S_i$.

for $\bar{n}$ iterations **do**

(2) Order the routes in S_i randomly.

for All pairs of routes in S_i **do**

(3) Try to improve the current pair of routes in S_i using GENICROSS.

(4) Try to improve both individual routes in the current pair of routes in S_i using IOPT.

end for

(5) Lower the threshold by $\Delta t : t = t - \Delta t$

(6) If a new best solution is found, store it: $S_b = S_i$. If no improvement has been found for $\bar{k}$ iterations, restart the search from the best solution obtained so far at a re-initialized threshold: $S_i = S_b$ and $t = t_{\max}$. If $t = 0$ for four iterations, re-initialize the threshold $t = t_{\max}$.

end for

(7) Return the best solution obtained, S_b.

is reduced by $\Delta t = 0.025$ units in each iteration until zero is reached. If the stopping criterion, the maximum number of iterations is not met, the threshold is reset to the maximum value ($t = t_{\max}$) and gradual lowering of the threshold is repeated for the remaining iterations. During the local search (Steps 3 and 4) the current best solution found is maintained in memory. If no improvement has been found for a certain number of iterations ($\bar{k} = 45$), we try improving the best solution with the threshold reset to its maximum.

In Step 6, the search is repeated for four iterations with $t = 0$ instead of immediately setting $t = t_{\max}$. This is because in each iteration at most one move to a neighboring solution is performed for each pair of routes, and each accepted move may trigger other improvements. Thus, several iterations are often required to reach the local optimum. The initial solutions are created using the MSLS algorithm described in the previous section. For exploring the neighboring solutions we use the IOPT-operator, introduced in Bräysy (2001) and a new improvement heuristic, GENICROSS. IOPT is a modification of well-known CROSS exchanges (Taillard et al., 1997), where also inversion of the order of the customers in the currently selected segment is considered. Here it is used only for intra-route optimization as inter-route exchanges are handled by GENICROSS.

GENICROSS draws upon well-known GENIUS insertion heuristic (Gendreau et al., 1992) and CROSS-exchanges (Taillard et al., 1997). The basic idea of CROSS-exchanges is to exchange segments of consecutive customers between routes. In CROSS-exchanges an attempt is made to insert segment S_2 from route R_2 only to a position in route R_1, where segment S_1 was removed, instead of trying all possible locations for S_2 in R_1. The GENICROSS operator evaluates all possible exchanges of segments between two routes R_1 and R_2. The maximum segment length is set to 5. Contrary to CROSS-exchanges, GENICROSS uses a first-accept strategy, and also considers moves in which the order of the customers in the selected segments is inverted. As this increases the complexity of the algorithm significantly, several speedup techniques are used.

The cost of the exchange is evaluated first and feasibility is checked only if an improvement is found. To further speed up evaluating the cost of moves, an exchange of segments S_1 and S_2 between routes R_1 and R_2 is divided in two insertions: S_1 from R_1 to R_2 and S_2 from R_2 to R_1. It is obvious that in case of improvement one or both of these segment relocations must improve the objective value. Therefore, we consider inserting S_2 to R_1 only if insertion of S_1 to R_2 improved the objective function value. This does not affect the worst case complexity, but makes the search a lot faster in practice.

To evaluate all possible moves, the GENICROSS operator must be performed twice, inverting the role of the routes. The limitation of this approach is that it considers only insertions between pairs of consecutive customers in the current routes. However, in the case of simultaneous exchanges between two routes it is possible that the insertion position of S_1 in route R_2 belongs to segment S_2 removed previously from R_2. In this case the actual insertion of S_1 will be between customers before and after the removed segment, preventing an insertion between a pair of consecutive customers. So, as in the GENIUS insertion heuristics of Gendreau et al. (1992), we consider here also insertions between pairs of customers that are not consecutive. The same speed-up techniques as described in Section 2 for the MSLS heuristics are used also here.

4. Computational experiments

We conducted an extensive computational study in order to assess the proposed approach with respect to robustness, scalability, and performance. The goals were to investigate our basic approach (denoted MSLS), to assess the effects of adding post-optimization (combined approach denoted MSLS + TA), and to compare both variants with state-of-the-art approaches reported in the literature.

4.1. Experimental design

For the experimental investigation, we used a set of 358 well-known benchmark problems from the literature. The set consists of the 56 100-customer problem instances of Solomon (1987), 300 benchmarks of Gehring and Homberger (1999)

and two real-life problems of 417 customers by Russell (1995).

The benchmarks of Solomon (1987) consists of 56 instances with 100 customers located in a 100×100 unit plane. The benchmark set contains six different subsets called R1, R2, RC1, RC2, C1, and C2. Locations of customers are uniformly distributed in R1 and R2, and clustered in C1 and C2. For groups RC1 and RC2, the clustered and random distributions are mixed. Furthermore, for instances of type 1, the capacity of the vehicle is small and the time window at the depot is narrow. Hence, only a few customers can be served by one vehicle, and relatively many vehicles are needed. Conversely, for instances of type 2, the vehicle capacity and time windows are less constraining. Hence more customers can be served by one vehicle, and fewer vehicles are needed.

Gehring and Homberger (1999) constructed similar sets of problem instances of 200, 400, 600, 800, and 1000 customers, with 60 instances in each set. The two problem instances (called D417 and E417) of Russell (1995) are extracted from a fast food routing application in southeastern United States. There are 417 customers in both problems. The two problems differ only in that the problem E417 has a higher percentage of tight time windows. For all 358 instances, distances and travel times between customers are measured by Euclidean distance. Each customer (including depot) has one time window for accepting service, an amount of requirement, and a service time. For these instances, minimizing the number of vehicles is considered the primary objective. For the same number of vehicles, the total travelled distance (or total duration) is often used as the secondary objective in the literature. In our methodological design, and the corresponding computational study, total distance is regarded as the secondary objective. The algorithms were coded in C++ and the computational experiments were conducted on a 700 MHz PC with 128 MB of RAM under the Windows 98 operating system.

4.2. Reliability and robustness

Because the MSLS heuristic contains stochastic elements, the reliability of the approach is an important property. To illustrate the performance of the MSLS and MSLS + TA approaches, Table 1 lists, to the best of our knowledge, all VRPTW heuristics for which best, average (or median), and worst solutions to Solomon's (1987) benchmark problems have been reported. In the lower part of the table, the results obtained by the MSLS and MSLS + TA approaches over 30 independent runs are reported. Table 1 presents both the number of routes and the total distance, averaged with respect to each problem group.

Comparing the approaches in Table 1 on their reliability is hard because only solutions with the same number of routes can be compared on their total distance. The MSLS, even without post-optimization, found higher quality solutions for the R1, R2, RC1 and RC2 problem sets than the other approaches listed in Table 1. For C1 and C2, the MSLS is on par with its competitors.

The small variance in number of vehicles and total distance between the best, average and worst results obtained for each of the problem instances, illustrates that our approach is reliable. The MSLS and MSLS + TA approaches are also robust: they obtain high quality results for the different problem instances. This can be shown by comparing the results of the MSLS and MSLS + TA to the other approaches in Table 1 or with the best approaches from the literature summarized in Table 3. Nevertheless, both approaches remain dependent on the quality of the initial solution. Table 1 demonstrates that the TA post-processor improves all MSLS results significantly, but the solutions obtained from the worst MSLS results always remain inferior to the ones obtained from the best MSLS results.

4.3. Scalability and adaptive search strategy

To investigate scalability, i.e., the relation between the CPU time and the problem size, we calculated the average CPU time over all Solomon (1987) and Gehring and Homberger (1999) cases.

For a fixed parameter setting for the 100, 200, 400, 600, 800, and 1000 customer cases by Solomon (1987) and Gehring and Homberger (1999), we observed that computing time rapidly grew with the number of customers. This observation

Table 1
Comparison of the best, average (or median) and worst results for Solomon's (1987) benchmarks

Author		R1	R2	C1	C2	RC1	RC2
1. Rochat and Taillard (1995)	Best	12.67	3.36	10.00	3.00	12.25	4.00
		1204.76	988.43	830.32	591.43	1385.28	1166.78
	Average	12.92	3.62	10.00	3.00	12.77	4.18
		1222.36	990.09	836.87	610.28	1418.36	1244.77
	Worst	13.17	3.91	10.00	3.00	13.25	4.38
		1257.72	1004.07	858.75	653.38	1470.99	1417.02
2. Taillard et al. (1997)	Best	12.25	3.00	10.00	3.00	11.88	3.38
		1216.70	995.38	828.45	590.30	1367.51	1165.62
	Average	12.33	3.00	10.00	3.00	11.90	3.38
		1220.35	1013.35	828.45	590.91	1381.31	1198.63
	Worst	12.50	3.00	10.00	3.00	12.00	3.38
		1219.71	1031.33	828.45	592.57	1402.12	1239.14
3. Liu and Shen (1999)	Best	12.17	2.82	10.00	3.00	11.88	3.25
		1249.57	1016.58	830.06	591.03	1412.87	1204.87
	Average	12.25	2.82	10.00	3.00	12.00	3.25
		1253.68	1022.08	841.33	591.03	1416.11	1230.31
	Worst	–	–	–	–	–	–
		–	–	–	–	–	–
4. Bent and Van Hentenryck (in press)	Best	12.17	2.73	10.00	3.00	11.63	3.25
		1203.84	980.31	828.38	589.86	1379.03	1158.91
	Average	12.25	2.85	10.00	3.00	11.80	3.33
		1208.40	986.90	828.38	609.39	1370.72	1167.24
	Worst	12.58	3.09	10.00	3.00	11.88	3.38
		1202.25	984.16	828.38	658.72	1374.21	1186.57
5. Homberger and Gehring (in press)	Best	12.17	2.82	10.00	3.00	11.75	3.25
		1217.77	959.41	830.22	592.04	1398.81	1144.91
	Average	12.17	2.85	10.00	3.00	11.83	3.25
		1217.79	959.84	832.24	592.12	1383.70	1158.05
	Worst	12.17	2.91	10.00	3.00	11.88	3.25
		1218.65	953.45	833.10	592.43	1371.99	1165.13
6. Bräysy et al. (in press)	Best	12.17	2.73	10.00	3.00	11.75	3.25
		1208.57	971.44	828.38	589.86	1372.93	1154.04
	Median	12.17	2.73	10.00	3.00	11.88	3.25
		1215.23	975.93	828.38	589.86	1380.55	1170.85
	Worst	12.17	2.73	10.00	3.00	12.00	3.25
		1230.54	997.85	828.38	589.86	1386.22	1203.97
7. MSLS (this paper)	Best	12.00	2.73	10.00	3.00	11.50	3.25
		1222.55	968.77	828.38	589.86	1400.91	1139.51
	Average	12.11	2.78	10.00	3.00	11.70	3.25
		1232.18	982.47	828.44	590.31	1403.34	1162.43
	Worst	12.25	2.91	10.00	3.00	12.00	3.25
		1244.78	986.37	828.61	591.73	1394.96	1195.42
8. MSLS + TA (this paper)	Best	12.00	2.73	10.00	3.00	11.50	3.25
		1212.89	960.44	828.38	589.86	1389.20	1124.14
	Average	12.11	2.78	10.00	3.00	11.70	3.25
		1218.56	967.19	828.38	589.86	1389.68	1146.61
	Worst	12.25	2.91	10.00	3.00	12.00	3.25
		1222.86	961.40	828.38	589.86	1378.71	1177.62

(1) Results obtained without the diversification and intensification scheme, Silicon Graphics 100 MHz, 5 runs, 92.2 (138) min, (2) Sun Sparc 10, 5 runs, 248 (248) min, (3) HP 9000/720, 3 runs, 20 (34) min, (4) Sun Ultra 10, 5 runs, 30 (318) min, (5) Pentium 400 MHz, 3 runs, 17.3 (93) min, (6) AMD 700 MHz, 3 runs, 14.1 (192) min, (7) AMD 700 MHz, 30 runs, 2.2 (30) min, (8) AMD 700 MHz, 30 runs, 2.6 (37) min.

motivated our development of an adaptive search strategy that will limit search effort as the problem size grows, without heavily compromising solution quality. The adaptive search strategy adjusts the values of some control parameters according to problem size.

First, within the inter-route exchanges of our local search, only the subset of 30 customers that are closest to the customers on the other route are considered for relocation. For each segment, only the 30 geographically closest insertion places are considered. This mainly affects the computational effort only in type 2 problems that are often more time consuming from the viewpoint of distance minimization. The other parameter adjustments are detailed in Table 2, where we describe for each problem size the number of initial solutions created, percentage of shortest routes considered for elimination, and number of iterations in the post-optimization phase, respectively.

The number of initial solutions is controlled by limiting the values tried for α_3 and the number of repetitions for the same values of α_1 and α_3 to try also different seed selection schemes. The lower and upper bounds for α_3 remained the same all the time (see Section 2.1), but in case of 200–600 customer problems we increment the value of α_3 by 0.3 units (instead of 0.2) and for larger problems by 0.5 units. Thus, for example in case of 800 customer problems, we try values 0.5, 1 and 1.5 for α_3 and values 0.6–1 in increments of 0.1 units for α_1. Thus, we try 3 values for α_3 and 5 values for α_1, i.e., 15 combinations. Then, all 15 combinations are tried twice leading to the total 30 initial solutions. The values for other parameters are described in Sections 2 and 3. The larger the problem instance, the more the adaptive search strategy limits the search efforts to keep computation times low. This slightly reduces the power of the MSLS + TA approach and limits its use for problem instances with more than 1000 customers.

Table 2
The parameter value adjustment for the large-scale test problems

Parameter/size	200	400	600	800	1000
Initial solutions	200	100	50	30	15
% Shortest routes	60	45	30	15	10
TA iterations	300	200	150	100	50

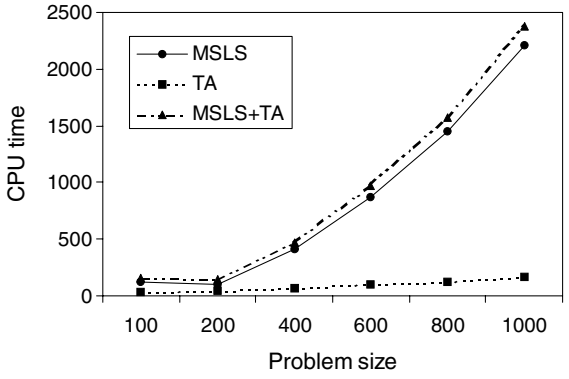

Fig. 1. Scalability of MSLS and MSLS + TA.

Three curves in Fig. 1 correspond to the CPU time of local search alone (MSLS), local search with post-optimization (MSLS + TA), and the isolated CPU time of post-optimization (TA) under the adaptive search strategy. The computation time needed for the MSLS heuristic clearly grows with problem size. The TA post-processor scales better as its computational effort increases moderately as problem instances become larger. The computational requirements of MSLS and MSLS + TA are compared to state-of-the-art approaches in the next subsection.

4.4. Computational results

The results from our experimental investigation on the 356 benchmark problems originating from Solomon (1987) and Gehring and Homberger (1999) are shown in Tables 3–8 below in a uniform way. Each table presents results for a given problem size. The local search (MSLS), and local search plus post-optimization (MSLS + TA) approaches, both with adaptive search, are compared with state-of-the-art results from the literature. We have included (to the best of our knowledge) all the best performing approaches for which the number of vehicles, total distance and computational effort have been reported. The first column gives the reference. Columns R1, R2, C1, C2,

Table 3
100 customer benchmark problems

Reference	R1	R2	C1	C2	RC1	RC2	CPU
Rochat and Taillard (1995)	12.58	3.09	10.00	3.00	12.38	3.62	Silicon Graphics 100 MHz,
	1197.42	954.36	828.45	590.32	1369.48	1139.79	1 run, 92.2 **(138)** min
Taillard et al. (1997)	12.25	3.00	10.00	3.00	11.88	3.38	Sun Sparc 10,
	1216.70	995.38	828.45	590.30	1367.51	1165.62	5 runs, 248 **(1240)** min
Homberger and Gehring (1999)	11.92	2.73	10.00	3.00	11.63	3.25	Pentium 200 MHz,
	1228.06	969.95	828.38	589.86	1392.57	1144.43	10 runs, 13 **(312)** min
Gehring and Homberger (1999)	12.42	2.82	10.00	3.00	11.88	3.25	4×Pentium 200 MHz,
	1198.00	947.00	829.00	590.00	1356.00	1144.00	1 run, 5 **(48)** min
Liu and Shen (1999)	12.17	2.82	10.00	3.00	11.88	3.25	HP 9000/720,
	1249.57	1016.58	830.06	591.03	1412.87	1204.87	3 runs, 20 **(102)** min
Gambardella et al. (1999)	12.38	3.00	10.00	3.00	11.92	3.33	Sun Ultrasparc 1,
	1210.83	960.31	828.38	591.85	1388.13	1149.28	1 run, 30 **(210)** min
Rousseau et al. (2002)	12.08	3.00	10.00	3.00	11.63	3.38	Sun Ultra 10,
	1210.21	941.08	828.38	589.86	1382.78	1105.22	10 runs, 183.3 **(19430)** min
Gehring and Homberger (2001)	12.00	2.73	10.00	3.00	11.50	3.25	4×Pentium 400 MHz,
	1217.57	961.20	828.63	590.33	1395.13	1139.37	5 runs, 13.5 **(1458)** min
Bräysy (2001)	11.92	2.73	10.00	3.00	11.50	3.25	Pentium 200 MHz,
	1222.12	975.12	828.38	589.86	1389.58	1128.38	1 run, 82.5 **(198)** min
Berger et al. (2003)	12.17	2.73	10.00	3.00	11.75	3.25	Pentium 400 MHz,
	1230.22	1009.53	828.48	589.93	1397.63	1230.20	3 runs, 30 **(486)** min
Ibaraki et al. (in press)	12.00	2.73	10.00	3.00	n.a.	3.25	Pentium 800 MHz,
	1216.03	960.29	828.38	589.86	n.a.	1129.44	1 run, 176 **(1705)** min
Bent and Van Hentenryck (in press)	12.17	2.73	10.00	3.00	11.63	3.25	Sun Ultra 10,
	1203.84	980.31	828.38	589.86	1379.03	1158.91	5 runs, 30 **(1590)** min
Li et al. (2003)	12.08	2.91	10.00	3.00	11.75	3.25	Pentium 545 MHz,
	1215.14	953.43	828.38	589.86	1385.47	1142.48	3 runs, 30 **(594)** min
Homberger and Gehring (in press)	12.08	2.82	10.00	3.00	11.50	3.25	Pentium 400 MHz,
	1211.67	950.72	828.45	589.96	1395.93	1135.09	5 runs, 17.5 **(473)** min
Bräysy and Dullaert (2003)	12.00	2.73	10.00	3.00	11.50	3.25	Pentium 700 MHz,
	1220.14	977.57	828.38	589.86	1397.44	1140.06	3 runs, 9.1 **(374)** min
MSLS (3 runs)	12.00	2.73	10.00	3.00	11.50	3.25	AMD 700 MHz,
	1235.22	979.88	828.38	589.86	1413.50	1152.37	3 runs, 2.1 **(86)** min
MSLS (30 runs)	12.00	2.73	10.00	3.00	11.50	3.25	AMD 700 MHz,
	1222.55	968.77	828.38	589.86	1400.91	1139.51	30 runs, 2.2 **(901)** min
MSLS + TA (3 runs)	12.00	2.73	10.00	3.00	11.50	3.25	AMD 700 MHz,
	1220.20	970.38	828.38	589.86	1398.76	1139.37	3 runs, 2.6 **(106)** min
MSLS + TA (30 runs)	12.00	2.73	10.00	3.00	11.50	3.25	AMD 700 MHz,
	1214.69	960.44	828.38	589.86	1389.20	1124.14	30 runs, 2.7 **(1106)** min

RC1, and RC2 present the average number of routes and average total distance with respect to the six problem groups of Solomon (1987). The rightmost column describes the computer, number of independent runs, and the CPU time used. For each approach, two CPU times are given. First, the one reported by the authors. Second (in the parentheses), a modified CPU time, where the computing times on various computers are scaled to equal Sun Sparc 10, using factors of Dongarra (1998). Moreover, if the results are the best ones over multiple independent runs, the computation times are multiplied by this number to illustrate the total computational effort. The reader must note that the computation times presented in the parenthesis are only indicative, and should be used

Table 4
200 customer benchmark problems

Reference	R1	R2	C1	C2	RC1	RC2	CPU
Gehring and Homberger (1999)	18.20	4.00	18.90	6.00	18.00	4.30	4×Pentium 200 MHz,
	3705	3055	2782	1846	3555	2675	1 run, 10 **(96)** min
Gehring and Homberger (2001)	18.20	4.00	18.90	6.00	18.10	4.40	4×Pentium 400 MHz,
	3855.03	3032.49	2842.08	1856.99	3674.91	2671.34	3 runs, 2.1 **(136)** min
Homberger and Gehring (in press)	18.20	4.10	19.00	6.00	18.10	4.50	Pentium 400 MHz,
	3890.06	3059.78	2836.66	1898.44	3734.32	2640.94	3 runs, 1.6 **(26)** min
Bent and Van Hentenryck (in press)	18.20	4.10	18.90	6.00	18.00	4.50	n.a.
	3677.96	3023.62	2726.63	1860.17	3279.99	2603.08	
Li et al. (2003)	18.30	4.10	19.10	6.00	18.30	4.90	Pentium 545 MHz,
	3736.20	3023.00	2728.60	1854.90	3385.80	2518.70	3 runs, 182.1 **(3606)** min
MSLS	18.20	4.00	18.90	6.00	18.00	4.40	AMD 700 MHz,
	3884.95	3081.61	2791.15	1860.71	3543.36	2672.01	3 runs, 1.7 **(68)** min
MSLS + TA	18.20	4.00	18.90	6.00	18.00	4.40	AMD 700 MHz,
	3718.30	3014.28	2749.83	1842.65	3329.62	2585.89	3 runs, 2.4 **(98)** min

Table 5
400 customer benchmark problems

Reference	R1	R2	C1	C2	RC1	RC2	CPU
Gehring and Homberger (1999)	36.40	8.00	38.00	12.00	36.10	8.60	4×Pentium 200 MHz,
	8925	6502	7584	3935	8763	5518	1 run, 20 **(192)** min
Gehring and Homberger (2001)	36.40	8.00	38.00	12.00	36.10	8.80	4×Pentium 400 MHz,
	9478.22	6650.28	7855.82	3940.19	9294.99	5629.43	3 runs, 7.1 **(460)** min
Homberger and Gehring (in press)	36.40	8.00	38.10	12.00	36.10	9.20	Pentium 400 MHz,
	9547.86	6683.53	7921.19	4049.71	9296.75	5609.88	3 runs, 5.1 **(83)** min
Bent and Van Hentenryck (in press)	36.40	8.00	38.00	12.00	36.10	8.90	n.a.
	8713.37	6959.75	7220.96	4154.40	8330.98	5631.70	
Li et al. (2003)	36.60	8.00	38.70	12.10	36.50	9.50	Pentium 545 MHz,
	8912.40	6610.60	7181.40	4017.10	8377.90	5466.20	3 runs, 359.8 **(7124)** min
MSLS	36.40	8.00	37.90	12.00	36.00	8.90	AMD 700 MHz,
	9225.95	6690.15	7464.09	3984.57	8836.49	5692.33	3 runs, 6.8 **(276)** min
MSLS + TA	36.40	8.00	37.90	12.00	36.00	8.90	AMD 700 MHz,
	8692.17	6382.63	7230.48	3894.48	8305.55	5407.87	3 runs, 7.9 **(322)** min

only to get a rough picture of the effort required to get the reported results.

Table 3 shows that MSLS + TA generates competitive results for the 100-customer problem instances. Based on 3 runs, it is faster than the main competitors and its results are within 0.36% from the best-known on average. To generate a single solution, the MSLS + TA spends on average the same amount of time on route reduction as on distance minimization. During the experiments route reduction is attempted for each solution in Phase 1. Distance minimization is only applied to solutions having the minimum number of routes (Phase 2) and to the best solution from Phase 2 during the TA post-optimization procedure. As a result, on average, 2/3 of the reported CPU time is spent on route reduction and 1/3 on distance minimization.

Increasing the number of runs to 30, allows the MSLS + TA approach to further reduce distance for problem groups R1, R2, RC1 and RC2, but reduces its speed advantage. Creating more initial

Table 6
600 customer benchmark problems

Reference	R1	R2	C1	C2	RC1	RC2	CPU
Gehring and Homberger (1999)	54.50 20 854	11.00 13 335	57.90 14 792	17.90 7787	55.10 18 411	11.80 11 522	4×Pentium 200 MHz, 1 run, 30 **(288)** min
Gehring and Homberger (2001)	54.50 21 864.47	11.00 13 656.15	57.70 14 817.25	17.80 7889.96	55.00 19 114.02	11.90 11 670.29	4×Pentium 400 MHz, 3 runs, 12.9 **(836)** min
Homberger and Gehring (in press)	54.50 21 605.60	11.00 13 682.21	57.90 15 086.01	18.00 7897.59	55.20 19 108.14	12.20 11 649.75	Pentium 400 MHz, 3 runs, 10.3 **(167)** min
Bent and Van Hentenryck (in press)	55.00 19 308.62	11.00 14 855.43	57.80 14 357.11	17.80 8259.04	55.10 17 035.91	12.40 11 987.89	n.a.
Li et al. (2003)	55.20 19 744.80	11.10 13 592.40	58.20 14 267.30	18.20 8202.60	55.50 17 320.00	13.00 11 204.90	Pentium 545 MHz, 3 runs, 399.8 **(7916)** min
MSLS	54.50 20 218.52	11.00 13 821.46	57.80 14 545.39	18.00 7918.32	55.00 17 953.54	12.10 11 993.25	AMD 700 MHz, 3 runs, 14.6 **(597)** min
MSLS + TA	54.50 19 081.18	11.00 13 054.83	57.80 14 165.90	18.00 7528.73	55.00 16 994.22	12.10 10 241.35	AMD 700 MHz, 3 runs, 16.2 **(661)** min

Table 7
800 customer benchmark problems

Reference	R1	R2	C1	C2	RC1	RC2	CPU
Gehring and Homberger (1999)	72.80 34 586	15.00 21 697	76.70 26 528	24.00 12 451	72.40 38 509	16.10 17 741	4×Pentium 200 MHz, 1 run, 40 **(384)** min
Gehring and Homberger (2001)	72.80 34 653.88	15.00 21 672.85	76.10 26 936.68	23.70 11 847.92	72.30 40 532.35	16.10 17 941.23	4×Pentium 400 MHz, 3 runs, 23.2 **(1503)** min
Homberger and Gehring (in press)	72.80 34 976.51	15.00 22 055.78	76.70 26 563.59	24.10 12 018.64	72.50 38 070.24	16.20 17 980.04	Pentium 400 MHz, 3 runs, 18.2 **(295)** min
Bent and Van Hentenryck (in press)	72.70 33 337.91	15.00 24 554.63	76.10 25 391.67	24.40 14 253.83	73.00 30 500.15	16.60 18 940.84	n.a.
Li et al. (2003)	73.00 33 806.34	15.10 21 709.39	77.40 25 337.02	24.40 11 956.60	73.20 31 282.54	17.10 17 561.22	Pentium 545 MHz, 3 runs, 512.9 **(10155)** min
MSLS	72.80 34 642.71	15.00 22 426.39	76.30 25 729.99	24.20 12 316.53	73.00 31 757.39	16.30 18 763.66	AMD 700 MHz, 3 runs, 24.1 **(983)** min
MSLS + TA	72.80 32 748.06	15.00 21 170.15	76.30 25 170.88	24.20 11 648.92	73.00 30 005.95	16.30 17 686.65	AMD 700 MHz, 3 runs, 26.2 **(1069)** min

solutions in a single run would have the same effect as doing several independent runs with less initial solutions. The effect of different values for the threshold and the effect of longer running times for the TA post-processor are described in Bräysy et al. (in press). Individual results on the Solomon (1987) problem instances are illustrated in the Appendix A. The figures in bold represent the instances for which the MSLS + TA approach matches best-known results.

The main strength of the MSLS + TA approach lies in solving larger problem instances. As Table 4 shows for the 200-customer problems, the MSLS + TA dominates the other approaches for the R2 and C2 subclasses. For the remaining subclasses, MSLS + TA is among the best. Homberger and Gehring (in press) is faster, but it does not perform as well. In the 400-customer benchmark cases, the MSLS + TA dominates the other approaches for all subclasses except RC2, as

Table 8
1000 customer benchmark problems

Reference	R1	R2	C1	C2	RC1	RC2	CPU
Gehring and Homberger (1999)	91.90 57 186	19.00 31 930	96.00 43 273	30.20 17 570	90.00 50 668	19.00 27 012	4×Pentium 200 MHz, 1 run, 50 **(480)** min
Gehring and Homberger (2001)	91.90 58 069.61	19.00 31 873.62	95.40 43 392.59	29.70 17 574.72	90.10 50 950.14	18.50 27 175.98	4×Pentium 400 MHz, 3 runs, 30.1 **(1950)** min
Homberger and Gehring (in press)	91.90 57 072.15	19.00 32 320.68	96.10 43 524.95	29.90 17 566.99	90.10 51 337.25	18.90 27 059.89	Pentium 400 MHz, 3 runs, 30.7 **(497)** min
Bent and Van Hentenryck (in press)	92.80 51 193.47	19.00 36 736.97	95.10 42 505.35	30.30 18 546.13	90.20 48 634.15	19.40 29 079.78	n.a.
Li et al. (2003)	92.70 50 990.80	19.00 31 990.90	96.30 42 428.50	30.80 17 294.90	90.40 48 892.40	19.80 26 042.30	Pentium 545 MHz, 3 runs, 606.3 **(12005)** min
MSLS	92.10 54 200.44	19.00 32 937.11	95.80 42 905.15	30.60 17 778.87	90.00 49 972.74	19.00 27 627.25	AMD 700 MHz, 3 runs, 36.8 **(1500)** min
MSLS + TA	92.10 50 025.64	19.00 31 458.23	95.80 42 086.77	30.60 17 035.88	90.00 46 736.92	19.00 25 994.12	AMD 700 MHz, 3 runs, 39.6 **(1616)** min

shown in Table 5. Dominance is only due to lower total distance, except for problem class RC1, where the number of vehicles is also lower. For RC2, Gehring and Homberger (1999) is superior. Homberger and Gehring (in press) and Gehring and Homberger (1999) are faster, but they do not dominate our approach. For the R1, R2, and RC1 subclasses of the 600-customer problem instances (Table 6), MSLS + TA clearly outperforms the other approaches due to lower distance values. For the C1, C2, and RC2 problems, other approaches are better. Gehring and Homberger (1999) and Homberger and Gehring (in press) are again faster, but neither dominates MSLS + TA. For the 800-customer problems, MSLS + TA only dominates for the R2 subclass, again due to lower distance value. For the remaining subclasses, our approach is outperformed, but not consistently by the same approach. Gehring and Homberger (1999) is faster and outperforms MSLS + TA for C2, RC1, and RC2. Table 8 for the 1000-customer instances shows that MSLS + TA outperforms all other approaches for the R2 subclass, although at the expense of a higher computational effort when compared with Gehring and Homberger (1999). For all other classes, the number of vehicles is competitive.

The results show that the MSLS approach produces highly competitive results for the main objective, i.e., number of vehicles, generally at a much lower computational cost than the competition. Post-optimization based on TA significantly improves the value of the secondary objective of total distance, at a very low marginal computational cost. For equal number of vehicles, the total distance results are highly competitive, and our method produces the best results known in a number of cases. Individual results on the Gehring and Homberger (1999) problem instances are presented in the Appendix A. The MSLS + TA results are particularly favorable for the 400 and 600 instances. Our approach reduced the total distance of 116 best-published solutions to the 300 instances of Gehring and Homberger (1999). Gehring and Homberger (1999) and Homberger and Gehring (in press) are the only approaches that scale better than MSLS + TA. From 400 customers upwards, these two approaches are faster. Still, they do not dominate our approach for any of the investigated problem sizes.

Table 9
Results for the two problem instances by Russell (1995)

Reference	CPU time (minutes)	Runs	D417		E417	
			Vehicles	Distance	Vehicles	Distance
Thangiah et al. (1994)	26	–	54	4866	55	4149
Kontoravdis and Bard (1995)	11	5	55	4273.40	55	4985.70
Rochat and Taillard (1995)	–	–	54	6264.80	54	7211.83
Russell (1995)	7	3	55	4964	55	6092
Chiang and Russell (1996)	25	–	55	4232.39	55	4397.49
Taillard et al. (1997)	–	–	55	3439.80	55	3707.10
Chiang and Russell (1997)	37	–	55	3455.28	55	3796.61
Liu and Shen (1999)	45	3	54	3747.52	54	4691.14
Homberger and Gehring (1999)	30	5	54	4703	55	4732
Gehring and Homberger (2001)	142	5	54	3512.01	54	3832.24
Bräysy (2001)	378	1	54	3506.21	54	3801.64
MSLS	45	3	54	3551.63	54	3927.40
MSLS + TA	47	3	54	3387.93	54	3672.73

The results for the two instances of Russell (1995) are shown in Table 9. Due to lack of information, we have not been able to produce a normalized CPU time for all references. Instead, we have listed CPU time and number of independent runs as they are presented in the references. The MSLS + TA approach improves the best published (as far as we know) results for the Russell (1995) problems, due to some 3% lower total distance. Based on the known information, our approach also seems to be at least as fast as the main competitors.

5. Conclusions

The VRPTW is a much-studied, computationally hard problem. It has many real-life applications, especially in the transportation industry. For industrial problems, scalable methods that are able to produce high quality results in a limited time, even for several hundreds of customers, are particularly important. With this focus, we have developed a novel, two-phase MSLS for the VRPTW. In the first phase, a sequential insertion heuristic is used to generate a number of initial solutions in a short time. Then all solutions are improved by a novel tour depletion approach. In the second phase, all solutions with the minimal number of routes are further improved with respect to distance by intra- and inter-tour operators. The best solution obtained after phase 2 can be post-optimized by two other distance reduction operators that are embedded in a TA metaheuristic search strategy. The new method has been investigated on 358 well-known test problem instances from the literature. Although it contains some stochastic elements, empirical investigation shows that the method is very robust and reliable. Using an adaptive search strategy, the method scales well up to 1000 customers. Speed performance is particularly good up to problem sizes up to some 500 customers. Comparisons with the best known results in the literature show that the method performs very favorable, and produces several new best-published results. The authors plan further investigations on scalable methods for large-size VRPTW in order to address industrial problems with several thousands of orders.

Acknowledgements

This work was partially supported by the Liikesivistysrahasto Foundation and the TOP program funded by the Research Council of Norway. This support is gratefully acknowledged. The authors are also grateful to both referees for their suggestions which improved the presentation of the paper.

Appendix A

Results for the benchmark problems of Solomon (1987)

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	01/4/1412.45
02/17/1486.12	02/3/1195.30	**02/10/828.94**	**02/3/591.56**	02/12/1558.07	02/3/1368.04
03/13/1293.69	03/3/944.55	**03/10/828.06**	**03/3/591.17**	03/11/1262.43	03/3/1061.16
04/10/985.33	04/2/838.56	**04/10/824.78**	**04/3/590.60**	04/10/1137.89	04/3/798.56
05/14/1377.11	05/3/1009.10	**05/10/828.94**	**05/3/588.88**	05/13/1642.53	05/4/1298.92
06/12/1253.23	06/3/913.24	**06/10/828.94**	**06/3/588.49**	06/11/1438.53	06/3/1152.14
07/10/1115.05	07/2/906.67	**07/10/828.94**	**07/3/588.29**	07/11/1233.30	07/3/1072.14
08/9/960.88	08/2/733.98	**08/10/828.94**	**08/3/588.32**	08/10/1143.42	08/3/829.69
09/11/1201.78	09/3/916.48	**09/10/828.94**			
10/10/1119.00	10/3/939.91				
11/10/1101.20	11/2/913.79				
12/9/1010.52					

Results for the benchmark problems of Gehring and Homberger (1999)

R1	R2	C1	C2	RC1	RC2
200					
01/20/4821.90	01/4/4648.89	01/20/2704.57	01/6/1931.44	01/18/4009.75	01/6/3177.87
02/18/4215.76	02/4/3776.70	02/18/3017.51	02/6/1867.00	**02/18/3364.23**	**02/5/2851.19**
03/18/3409.23	03/4/2998.56	03/18/2803.60	03/6/1814.94	03/18/3080.54	03/4/2645.70
04/18/3120.48	04/4/2039.08	04/18/2662.49	04/6/1755.32	**04/18/2922.80**	04/4/2110.22
05/18/4380.41	**05/4/3391.43**	05/20/2702.05	05/6/1879.31	05/18/3587.17	05/4/3105.87
06/18/3667.17	06/4/2974.83	06/20/2701.04	06/6/1858.12	06/18/3549.09	06/5/2674.11
07/18/3262.44	07/4/2551.49	07/20/2701.04	07/6/1849.46	07/18/3386.56	07/4/2659.55
08/18/3000.46	08/4/1880.88	08/19/2804.23	08/6/1826.02	08/18/3157.72	08/4/2352.27
09/18/3934.92	09/4/3154.57	09/18/2741.52	**09/6/1833.37**	**09/18/3148.18**	09/4/2248.58
10/18/3370.20	10/4/2726.40	**10/18/2660.29**	10/6/1811.53	10/18/3090.20	**10/4/2033.57**
400					
01/40/10629.89	01/8/9533.86	01/40/7153.10	01/12/4124.66	01/36/9657.15	01/12/6806.67
02/36/9425.63	02/8/7837.69	02/37/7501.54	02/12/3971.37	02/36/8317.46	02/10/6287.59
03/36/8028.00	**03/8/6218.64**	03/36/7302.08	03/12/3884.91	03/36/7820.79	**03/8/5276.97**
04/36/7503.39	**04/8/4409.32**	04/36/6953.08	**04/12/3722.63**	**04/36/7471.43**	04/8/3780.66
05/36/9877.32	**05/8/7376.67**	05/40/7152.06	05/12/3951.61	**05/36/8753.52**	05/10/5970.88
06/36/8692.75	**06/8/6377.94**	06/40/7154.76	06/12/3887.40	**06/36/8577.35**	06/9/5976.26
07/36/7802.09	**07/8/5254.74**	07/40/7149.44	07/12/3910.77	**07/36/8412.15**	07/8/5734.69
08/36/7466.31	**08/8/4193.88**	**08/38/7208.43**	08/12/3832.47	**08/36/8148.61**	08/8/5113.01
09/36/9115.36	**09/8/6604.69**	**09/36/7688.24**	09/12/3927.78	**09/36/8047.53**	**09/8/4693.48**
10/36/8380.96	**10/8/6018.84**	**10/36/7042.09**	10/12/3731.15	**10/36/7849.51**	10/8/4438.45

R1	R2	C1	C2	RC1	RC2
600					
01/59/22189.27	01/11/19329.52	01/60/14146.34	01/18/7780.84	**01/55/19014.86**	01/16/13667.92
02/54/20373.21	02/11/15462.55	02/56/14723.81	02/18/7748.42	**02/55/16816.09**	02/14/11591.25
03/54/17988.65	**03/11/11915.86**	**03/56/14098.96**	03/18/7436.36	**03/55/15902.07**	03/11/10671.06
04/54/16567.28	**04/11/8525.42**	**04/56/13845.64**	04/18/7210.07	**04/55/15376.27**	**04/11/7632.79**
05/54/21580.31	**05/11/15786.15**	05/60/14114.68	05/18/7618.26	**05/55/17899.56**	05/13/12982.88
06/54/19094.62	06/11/13305.31	06/60/14119.19	06/18/7512.39	**06/55/17798.51**	06/12/12486.91
07/54/17553.19	07/11/10868.46	07/60/14097.52	**07/18/7667.39**	**07/55/17188.86**	07/11/12098.58
08/54/16324.95	**08/11/8075.55**	08/58/14751.28	**08/18/7464.88**	**08/55/16777.26**	**08/11/10935.97**
09/54/20212.92	**09/11/14193.30**	**09/56/13977.47**	09/18/7568.16	**09/55/16724.39**	**09/11/10346.11**
10/54/18927.37	**10/11/13086.16**	**10/56/13784.09**	10/18/7280.48	**10/55/16444.31**	**10/11/9709.96**
800					
01/80/38540.40	01/15/29778.16	01/80/25299.79	01/24/11917.28	01/73/32953.99	01/21/21680.78
02/72/34358.80	02/15/23942.59	02/76/25734.81	02/24/11698.36	02/73/29625.85	02/18/18915.90
03/72/30711.02	**03/15/18792.94**	03/73/25041.87	03/25/11872.82	03/73/28477.15	03/15/15884.89
04/72/29089.37	**04/15/14029.92**	**04/72/24219.23**	04/24/11604.61	04/73/27584.57	04/15/12362.60
05/72/36362.38	05/15/25962.56	05/80/25265.89	05/24/11561.18	**05/73/31035.67**	05/18/19622.48
06/72/32412.67	**06/15/21660.17**	06/80/25293.71	06/24/11539.97	**06/73/30896.72**	06/16/20034.93
07/72/30041.23	**07/15/17563.25**	07/80/25234.76	07/25/11736.21	07/73/30492.24	07/15/18622.57
08/72/28793.35	**08/15/13887.83**	08/76/25737.39	08/24/11497.50	08/73/29849.56	08/15/17367.36
09/72/34527.50	09/15/23905.50	**09/73/25216.66**	**09/24/11688.26**	09/73/29756.55	**09/15/16739.85**
10/72/32643.85	10/15/22178.56	10/73/24664.68	10/24/11373.05	10/73/29387.23	**10/15/15635.17**
1000					
01/100/55998.03	01/19/45198.44	01/100/42676.12	01/30/17030.37	**01/90/50673.09**	01/23/31768.89
02/92/50633.40	02/19/36588.92	02/95/42509.69	02/31/17055.34	**02/90/46394.54**	02/21/26956.04
03/91/47466.88	**03/19/27162.51**	03/91/41564.72	03/31/17245.16	**03/90/44149.27**	**03/18/22239.43**
04/91/44743.90	**04/19/19551.15**	**04/90/40475.11**	04/30/17610.28	**04/90/43074.82**	**04/18/17039.80**
05/92/53338.29	05/19/39251.86	05/100/42514.23	05/31/16904.93	**05/90/48853.78**	05/20/29306.77
06/91/50438.84	06/19/32340.63	06/100/43038.82	06/30/16843.64	**06/90/48325.30**	06/18/29499.02
07/91/46680.31	07/19/25928.47	07/100/42529.55	07/32/17216.83	**07/90/47444.73**	**07/18/27533.09**
08/91/44392.65	08/19/19561.45	08/98/42451.59	08/30/16454.37	**08/90/46351.53**	**08/18/26122.10**
09/91/55275.32	09/19/36028.83	09/93/41948.19	09/31/17515.85	**09/90/46325.38**	**09/18/25193.30**
10/91/51288.75	10/19/32970.00	10/91/41159.70	10/30/16482.02	**10/90/45776.74**	**10/18 /24282.80**

References

Aarts, J., Korst, H., Van Laarhaven, P., 1997. Simulated annealing. In: Aarts, E., Lenstra, J. (Eds.), Local Search in Combinatorial Optimization. John Wiley & Sons, Chichester, pp. 91–120.

Bent, R., Van Hentenryck, P., in press. A two-stage hybrid local search for the vehicle routing problem with time windows. Transportation Science, in press.

Berger, J., Barkaoui, M., Bräysy, O., 2003. A route-directed hybrid genetic approach for the vehicle routing problem with time windows. Information Systems and Operations Research 41, 179–194.

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

Bräysy, O., 2001. Local search and variable neighborhood search algorithms for the vehicle routing problem with time

windows. Doctoral thesis. Department of Mathematics and Statistics, University of Vaasa, Finland.

Bräysy, O., Dullaert, W., 2003. A fast evolutionary metaheuristic for the vehicle routing problem with time windows. International Journal on Artificial Intelligence Tools 12, 153–172.

Bräysy, O., Gendreau, M., in press a. Vehicle routing problem with time windows, Part I: Route construction and local search algorithms. Transportation Science, in press.

Bräysy, O., Gendreau, M., in press b. Vehicle routing problem with time windows, Part II: Metaheuristics. Transportation Science, in press.

Bräysy, O., Berger, J., Barkaoui, M., Dullaert, W., in press. A threshold accepting metaheuristic for the vehicle routing problem with time windows. Central European Journal of Operations Research, in press.

Caseau, Y., Laburthe, F., Silverstein, G., 1999. A metaheuristic factory for vehicle routing problems. In: Jaffar, J. (Ed.), Principles and Practice of Constraint Programming—CP'99. In: Lecture Notes in Computer Science. Springer-Verlag, New York, pp. 144–158.

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

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

Cook, W., Rich, J., 1999. A parallel cutting-plane algorithm for the vehicle routing problems with time windows. Technical report, Department of Computational and Applied Mathematics, Rice University, Houston. Available from http://www.caam.rice.edu/caam/tr99.html.

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

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

Czech, Z., Czarnas, P. 2002. Parallel simulated annealing for the vehicle routing problem with time windows. In: Proceedings of 10th Euromicro Workshop on Parallel, Distributed and Network-Based Processing, Canary Islands, Spain, pp. 376–383.

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

Dueck, G., Scheurer, T., 1990. Threshold accepting: A general purpose optimization algorithm appearing superior to simulated annealing. Journal of Computational Physics 90, 161–175.

Dueck, G., Scheuer, T., Wallmeier, H.-M., 1993. Toleranzschwelle und sintflut: Neue ideen zur optimierung. Spektrum der Wissenschaft 3, 42–51.

Dullaert, W., Bräysy, O., in press. Routing relatively few customers per route. TOP, in press.

Gambardella, L., Taillard, E., Agazzi, G., 1999. MACS-VRPTW: A multiple ant colony system for vehicle routing problems with time windows. In: Corne, D., Dorigo, M., Glover, F. (Eds.), New Ideas in Optimization. McGraw-Hill, London, pp. 63–76.

Gehring, H., Homberger, J., 1999. A parallel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows. In: Miettinen, K., Mäkelä, M., Toivanen, J. (Eds.), Proceedings of EUROGEN99, University of Jyväskylä, Jyväskylä, Finland, pp. 57–64.

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

Gendreau, M., Hertz, A., Laporte, G., 1992. New insertion and postoptimization procedures for the travelling salesman problem. Operations Research 40, 1086–1093.

Glover, F., 1991. Multilevel tabu search and embedded search neighborhoods for the travelling salesman problem. Technical report, College of Business & Administration, University of Colorado, Boulder, CO.

Glover, F., 1992. New ejection chain and alternating path methods for traveling salesman problems. In: Balchi, O., Ramesh, S., Zenios, S.A. (Eds.), Computer Science and Operations Research: New Developments in Their Interfaces. Pergamon Press, Oxford, pp. 449–509.

Homberger, J., Gehring, H., 1999. Two evolutionary metaheuristics for the vehicle routing problem with time windows. Information Systems and Operational Research—Special issue: Metaheuristics for Location and Routing Problems 37, 297–318.

Homberger, J., Gehring, H. A two phase hybrid metaheuristic for the vehicle routing problem with time windows. European Journal of Operational Research, in press.

Ibaraki, T., Kubo, T., Uno, T., Yagiura, M., in press. Effective local search algorithms for the vehicle routing problem with general time windows. Transportation Science, in press.

Ioannou, G., Kritikos, M., Prastacos, G., 2001. A greedy look-ahead heuristic for the vehicle routing problem with time windows. Journal of the Operational Research Society 52, 523–537.

Kirkpatrick, S., Gelatt, C., Vecchi, M., 1983. Optimization by simulated annealing. Science 220, 671–680.

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

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

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

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

Lenstra, J., Rinnooy Kan, A., 1981. Complexity of vehicle routing and scheduling problems. Networks 11, 221–227.

Li, H., Lim, A., Huang, J., 2003. Local search with annealing-like restarts to solve the VRPTW. European Journal of Operational Research 150, 115–127.

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

Metropolis, W., Rosenbluth, A., Rosenbluth, M., Teller, A., Teller, E., 1953. Equation of the state calculations by fast computing machines. Journal of Chemical Physics 21, 1087–1092.

Potvin, J., Rousseau, J., 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., Rousseau, J.-M., 1995. An exchange heuristic for routing problems with time windows. Journal of the Operational Research Society 50, 1433–1446.

Rego, C., 1998. A subpath ejection method for the vehicle routing problem. Management Science 44, 1447–1459.

Rego, C., 2001. Node ejection chains for the vehicle routing problem: Sequential and parallel algorithms. Parallel Computing 27, 201–222.

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

Rousseau, L.-M., Gendreau, M., Pesant, G., 2002. Using constraint-based operators to solve the vehicle routing problem with time windows. Journal of Heuristics 8, 43–58.

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

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. In: Maher, M., Puget, J.-F. (Eds.), Principles and Practice of Constraint Programming—CP98. In: Lecture Notes in Computer Science. Springer-Verlag, New York, pp. 417–431.

Solomon, M., 1987. Algorithms for the vehicle routing and scheduling problem with time window constraints. Operations Research 35, 254–265.

Solomon, M., Baker, E., Schaffer, J., 1988. Vehicle routing and scheduling problems with time window constraints: Efficient implementations of solution improvement procedures. In: Golden, B., Assad, A. (Eds.), Vehicle Routing: Methods and Studies. Elsevier Science Publishers (North-Holland), Amsterdam, pp. 85–105.

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

Thangiah, S., Osman, I., Vinayagaoorthy, R., Sun, T., 1994. Algorithms for vehicle routing problems with time deadlines. American Journal of Mathematical and Management Science 13, 323–355.

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