Journal of Heuristics, 8: 251–276, 2002

Parallelization of a Two-Phase Metaheuristic for Routing Problems with Time Windows

HERMANN GEHRING AND JÖRG HOMBERGER
University of Hagen, Profilstraße 8, D-58084 Hagen, Germany

Abstract

This paper describes the parallelization of a two-phase metaheuristic for the vehicle routing problem with time windows and a central depot (VRPTW). The underlying objective function combines the minimization of the number of vehicles in the first search phase of the metaheuristic with the minimization of the total travel distance in the second search phase. The parallelization of the metaheuristic follows a type 3 parallelization strategy (cf. Crainic and Toulouse (2001). In F. Glover and G. Kochenberger (eds.). *State-of-the-Art Handbook in Metaheuristics*. Norwell, MA: Kluwer Academic Publishers), i.e. several concurrent searches of the solution space are carried out with a differently configured metaheuristic. The concurrently executed processes cooperate through the exchange of solutions. The parallelized two-phase metaheuristic was subjected to a comparative test on the basis of 358 problems from the literature with sizes varying from 100 to 1000 customers. The derived results seem to justify the proposed parallelization concept.

Key Words: evolution strategy, metaheuristic, tabu search, hybridization, parallelization, vehicle routing, time windows

1. Introduction and problem formulation

The vehicle routing problem with time windows (VRPTW) is an extension of the well-known vehicle routing problem (VRP). It can be described as follows (cf. Desrochers et al., 1988; Solomon and Desrosiers, 1988):

n customers are to be serviced from a depot with vehicles of the same capacity Q. Each customer $i, i = 1, \ldots, n$, is characterized by a demand q_i, a service time s_i and a service time window $z_i = [e_i, f_i]$, where e_i denotes the earliest and f_i the latest time for the start of servicing. The demand q_i of a customer i is to be covered by exactly one service within the time window z_i. In addition, e_0 describes the earliest time for the departure of a vehicle from the depot $i, i = 0$, and f_0 the latest time for the arrival at the depot. The locations of the depot and the customers and the shortest distances d_{ij} and corresponding travel times d'_{ij} between two locations are given. The objective is to determine a feasible solution which minimizes primarily the number of vehicles (primary criterion) and secondarily the total travel distance (secondary criterion).

The VRP and the VRPTW belong to the class of NP-hard combinatorial optimization problems (cf. Lenstra and Rinnooy Kan, 1981). However, the VRPTW in particular is still "much more difficult to solve in practice" than the VRP (cf. Solomon, Baker, and Schaffer, 1988). It is therefore no surprise when primarily heuristic procedures are suggested for larger problem instances of the VRPTW. In recent years, quite good results have been

achieved for the VRPTW with metaheuristics such as tabu search (see Potvin et al., 1996; Taillard et al., 1996; Chiang and Russell, 1997; Liu and Shen, 1999), simulated annealing (see Chiang and Russell, 1996), genetic algorithms (see Thangiah, Nygard, and Juell, 1991; Potvin and Bengio, 1996) and evolution strategies (see Homberger and Gehring, 1999). Only a few authors report on solutions to the VRPTW by means of hybrid and parallel approaches.

According to Glover, Kelley, and Laguna (1995) hybrid approaches focus on enhancing the strengths and compensating for the weaknesses of two or more complementary approaches. The intention is to generate better solutions by combining the key elements of competing methodologies. Following this idea, Thangiah, Osman, and Sun (1995) have proposed a two-phase hybrid metheuristic for the VRPTW. The best individual determined by a genetic algorithm in the first search phase is passed over to a tabu search algorithm which improves the received solution in the second search phase. The tabu search algorithm itself uses an element of the simulated annealing concept for controlling the selection of neighbourhood solutions. The hybrid from Chiang and Russell (1996) carries out two subsequent steps repeatedly. Partial routes are formed in the first step and improved by simulated annealing in the second step. In order to avoid cycling, a tabu list is used in the simulated annealing steps. The hybrid suggested by Gehring and Homberger (1999) also proceeds in two phases. First, the number of vehicles is minimized by a $(1, \lambda)$-evolution strategy and then a tabu search is used to minimize the total travel distance.

Parallel solution approaches may be motivated, e.g., by solving a larger problem in a given time (scale-up), generating a result in a shorter time (speed-up), calculating a better solution in a given time, improving convergence behaviour, etc. The emphasis is often on improving the solution quality. Reviews of the state-of-the-art in the field of parallel metaheuristics are given in Toulouse, Cranic, and Gendreau (1996) and Crainic and Toulouse (1998, 2001). According to Toulouse, Cranic, and Gendreau (1996), three types of parallelization strategies seem to be appropriate for the methods most used in combinatorial optimization: (1) parallelization of operations within an iteration of the solution method, (2) decomposition of problem domain or search space, and (3) multi-search threads with various degrees of synchronization and cooperation. Garcia, Potvin, and Rousseau (1994) use a type 1 parallelization for solving the VRPTW. Within an iteration of a tabu search method, neighbourhood solutions are generated and evaluated in parallel. Taillard (1993) developed a type 2 parallelization strategy for vehicle routing problems. The key idea of this approach is to decompose the domain into polar regions, to which vehicles are allocated, and to solve each subproblem by an independent tabu search. The resulting processes are synchronized after a certain number of iterations and the solutions of the subproblems are combined into a complete solution. The partition into subproblems is then modified. Schulze and Fahle (1999) solve the VRPTW by means of a type 3 parallelization. Several concurrent searches of the solution space are carried out with a differently configured tabu search algorithm. The search processes cooperate through the exchange of solutions. Another way of cooperation in the context of type 3 parallelization is used by Gambardella, Taillard, and Agazzi (1999), who propose a multiple ant colony system. Cooperation between artificial ant colonies is performed by exchanging information through pheromone updating.

In this paper a type 3 parallelization approach for solving the VRPTW is presented and evaluated. The multi-search threads are realized by a differently configured two-phase metaheuristic, which is a modification of the hybrid from Gehring and Homberger (1999) referred to above. In contrast to this method, an approximate version of which was presented by the authors at the Eurogen 99, a (μ, λ)-evolution strategy—instead of a $(1, \lambda)$-evolution strategy—is now used to minimize the number of vehicles in the first search phase. Further differences in the first search phase concern the evaluation of individuals passed over to the next generation and the criterion for terminating the first search phase. In the second search phase the total travel distance is again minimized with the same tabu search algorithm. However, another termination criterion for the second search phase is used here. In each of the search phases process cooperation by means of solution exchange takes place. Finally, the evaluation reported here is based on two additional test problems and on several runs per test instance instead of one run only.

Most of the metaheuristics which have been published so far are not (explicitly) tailored to larger instances of the VRPTW. An essential aim of this paper is therefore to contribute to the development of algorithms which are capable of solving instances with up to 1000 customers. To keep the computing time within reasonable limits, a parallel approach is chosen. Hence, parallel aspects of the solution approach, such as speed-up, synergetic effects, etc., are also considered.

The paper is organized as follows: Section 2 describes the underlying two-phase metaheuristic and Section 3 the parallelization of this two-phase metaheuristic. In Section 4 the performance of the parallel approach is demonstrated by means of a comparative test including methods from other authors as well. Finally, Section 5 contains some concluding remarks.

2. A two-phase hybrid metaheuristic for the VRPTW

A neighbourhood search controlled by a metaheuristic which simultaneously tries to reduce the number of vehicles and the total travel distance may suffer from specific difficulties. Since the distribution of the customers to be serviced over a smaller number of routes is heavily hampered by the time window constraints (cf. Gietz, 1994), the probability of reducing the number of vehicles in an iteration of a neighbourhood search is very low. Hence, the search process is, de facto, more or less guided by the secondary optimization goal and the primary criterion, the minimization of the number of vehicles, falls behind. If, for a given number of vehicles, the further search direction is determined exclusively through the secondary criterion, i.e. minimization of the total travel distance, the achievement of the primary criterion, i.e. minimization of the number of vehicles, may be impeded. Retzko (1995) shows that the total travel distance usually increases if the number of vehicles falls below a defined value. Vice versa, it appears plausible that minimization of the total travel distance does not inevitably lead to a reduction in the number of vehicles.

As a remedy a two-phase metaheuristic is proposed which explicitly considers the primary criterion. The metaheuristic also takes advantage of the hybridization concept and combines a (μ, λ)-evolution strategy and a subsequently executed tabu search. During the first phase, when the (μ, λ)-evolution strategy is applied, the search process is dedicated to the reduction of the number of vehicles. The best solution found in the first search phase is passed over

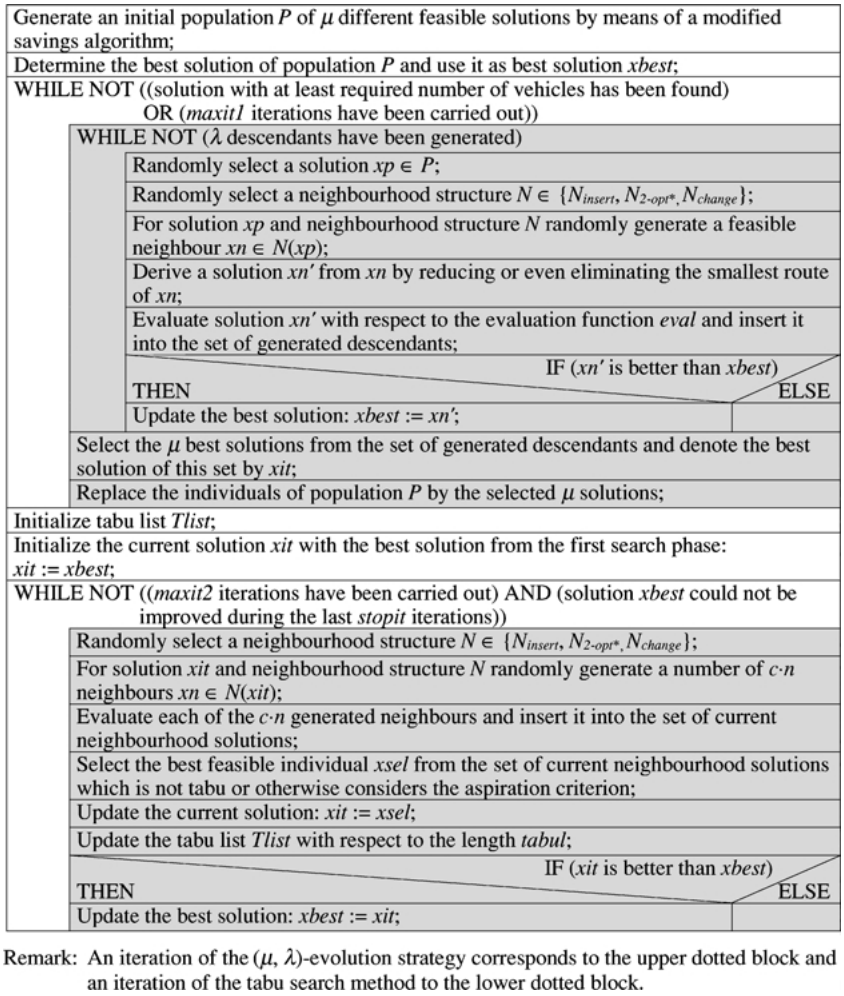

Figure 1. Structure chart for the two-phase hybrid metaheuristic.

to the second phase. No problems arise with the encoding of transferred solutions, since an identical encoding is used in both phases: solutions to the problem are represented by sequence vectors indicating the order in which the customers are served. In the second phase a tabu search is carried out. The goal pursued here is the minimization of the total travel distance, i.e. no attempts are made to reduce the number of vehicles further. The overall two-phase search process is roughly outlined in figure 1.

The first search phase starts with the generation of an initial population P of μ different feasible solutions. For this purpose a modification of the savings algorithm from Clarke and Wright (1964) is used. The modification comprises the imposed time window constraints (cf. Solomon, 1987) and the inclusion of a stochastic component in the iterated selection of the next savings element.

In each iteration of the (μ, λ)-evolution strategy $\lambda, \lambda > \mu$, descendants of the current population are generated and evaluated. The μ best descendants form the next population. In addition, the best solution is, possibly, updated in each iteration. The first phase terminates, if a solution with a minimum required number of vehicles is found, or if a given number *maxit1* of iterations of the (μ, λ)-evolution strategy has been carried out. The core operations of this procedure, the generation and the evaluation of descendants, are described in more detail below.

The generation of a descendant involves measures aiming at the reduction of the number of vehicles. A descendant is calculated in three steps:

In the first step a solution *xp* is randomly selected from the current population P. The selection probability is the same for each element of P.

The second step comprises the generation of a descendant candidate. First, a neighbourhood structure N is randomly selected from the set of neighbourhood structures $\{N_{insert}, N_{2-opt^*}, N_{change}\}$. According to Hansen and Mladenovic (1999), the change of the neighbourhood structure during the search achieved in this way has a positive impact on the search process. The neighbourhood structure N_{insert} is based on an exchange concept from Or (1976), while N_{2-opt^*} was introduced by Potvin and Rousseau (1995) and N_{change} by Osman (1993). A feasible move is then randomly determined for neighbourhood structure N and applied to solution *xp*. If the resulting descendant candidate *xn* is infeasible, the candidate generation is repeated. Each of the available move operators comprises a stochastic element. In the case of the neighbourhood structure N_{insert}, for example, the customer to be removed and the insertion position are randomly determined for the corresponding move operator. The repetition of a move therefore usually leads to a different descendant candidate.

The third step serves the transformation of candidate *xn* to the final neighbourhood solution xn' by means of a modified Or-opt operator (cf. Homberger and Gehring, 1999). Let the term "smallest route" denote the route of a solution containing the smallest number of customers. The modified Or-opt operator then aims at reducing the number of customers of the smallest route of solution *xn* or even at eliminating the smallest route. For this purpose, attempts are made subsequently to insert all customers of the smallest route in a feasible way into other routes. If no customer can be inserted into another route, solution *xn* is transferred unchanged to xn'.

For details of the second and the third step see Homberger and Gehring (1999).

The evaluation of a solution underlying the selection of the μ best descendants follows Homberger and Gehring (1999) and is based on the four criteria "total number of vehicles", "number of customers of the smallest route", "minimal delay" and "total travel distance". These criteria are applied in lexicographic order as given above. The so-called "minimal delay" refers to the smallest route of a solution and estimates how easily this route can be eliminated in the further search process (for details see Homberger and Gehring, 1999). The criterion "number of customers of the smallest route" pursues the same purpose. On the other hand, a second lexicographic evaluation function is used here alternatively. This is derived from the first evaluation function mentioned above by replacing the criterion "minimal delay" by the so-called "caused overload". The rationale of introducing the latter criterion is as follows.

With respect to the question of how easily the smallest route of a solution can be eliminated, the criterion "minimal delay" considers the imposed time window constraints only. The elimination of routes is undoubtedly hampered by these constraints. It is, however, also possible that in certain cases the elimination of a route is prevented by the vehicle capacity. The "caused overload" tries to estimate the impact of this limitation. Let *rs* denote a solution and *rmin*, $rmin \in rs$, the smallest route of this solution. In addition, let *rmax*, $rmax \in rs$, $rmax \neq rmin$, denote the route of solution *rs* with the largest still unused part of the vehicle capacity Q. The "caused overload" *ovl* then states the extent to which the respective vehicle is overloaded, if the customer with the smallest demand of the smallest route *rmin* is inserted into route *rmax*. The "caused overload" *ovl* is determined as follows:

$$ovl = \left| \min\left\{ \left(Q - \sum_{i \in rmax} q_i - \min_{k \in rmin} q_k \right), 0 \right\} \right|. \tag{1}$$

The two evaluation functions, the one with "minimal delay" and the other with "caused overload", are used to configure the concurrently executed search processes differently. In the following, the selection of one of these two evaluation functions is formalized by means of the parameter *eval* and the parameter values "mdel" for inclusion of criterion "minimal delay" and "ovld" for "caused overload". However, the current determination and update of the best solution *xbest*, which in the end is passed over to the second search phase, considers two evaluation criteria only: the number of vehicles as the primary criterion and the total travel distance as the secondary criterion.

In the second search phase, the tabu search concept from Glover is used. For a detailed description of the tabu search method see Glover (1989, 1990) and Glover and Laguna (1993). At the beginning, the tabu list *Tlist* is initialized as an empty list and the current solution is initialized with the best solution from the first search phase. In each of the following iterations of the tabu search method a number of $c \cdot n$ neighbours of the current solution, where c, $c > 0$, is a constant, is randomly generated and evaluated. The underlying neighbourhood structure is randomly selected from the same set of neighbourhood structures that was used in the first search phase. The evaluation of each of the neighbours is based on the number of vehicles as the primary criterion and the total travel distance as the secondary criterion. The best feasible evaluated neighbour is selected from the set of generated neighbours and used as the new current solution if it meets the following condition: the selected feasible neighbour is not tabu or otherwise respects the aspiration criterion.

A selected solution is not tabu if none of the connections between customers or between customers and the depot are set tabu in the tabu list. However, according to the aspiration criterion, the tabu status of one or more connections of a solution is ignored if the respective neighbour represents a new best solution. At the end of an iteration of the tabu search method, the tabu list *Tlist* and the best solution *xbest* are updated. The update of the tabu list concerns those connections between customers, or between customers and the depot, which are eliminated when the transition from the current solution to its best evaluated neighbour is performed (cf. Potvin et al., 1996). An eliminated connection is stored in the tabu list for *tabul* iterations. The second search phase is terminated if two criteria are met: (1) a number of *maxit2* iterations of the tabu search method has been carried out and (2) the best solution *xbest* could not be improved during the last *stopit*, $stopit > 0$, iterations.

3. Parallelization of the two-phase hybrid metaheuristic

The parallelization concept used here takes the available technological resources into account. Instead of a massive-parallel computer system only a PC-LAN is available. Hence, a coarse grained parallelization of type 3 is chosen. The first part of this section contains a conceptual view of the applied strategy and the actual implementation is introduced in the second part.

3.1. Conceptual view

The conceptual view of the strategy and collaboration is based on a taxonomy proposed by Crainic and Toulouse (2001). This taxonomy includes several "conceptual criteria" of varying strategic impact: heuristic methods per thread, starting points of search, distribution of communication with regard to time, synchronization of communication, and moments of communication. In the following, the design decisions corresponding to these criteria are described approximately and substantiated.

3.1.1. Heuristic methods per thread. Each concurrent thread may or may not execute the same heuristic method. In the first case the applied (same) method may be configured equally or differently. Here, each concurrent thread executes the same method or two-phase hybrid metaheuristic respectively. In order to enhance the chances of finding better solutions and to increase the robustness of the global search, the configuration of the method varies between the threads. On the one hand, the ratio of the search times available for the first and the second search phase is varied. A smaller value of the respective parameter *maxit1* causes earlier entry into the second search phase and thus supports the calculation of a better solution with respect to the total travel distance. Vice versa, a higher value enhances the chances of finding a better solution in terms of the number of vehicles. On the other hand, the robustness of the global search is increased by implementing different values of parameter *eval* ("mdel" or "ovld"). While "mdel" considers situations where the elimination of routes, i.e. the reduction of the number of vehicles, is hampered by the time window constraints, "ovld" provides for analogous impacts caused by the capacity constraints.

3.1.2. Starting points of search. Each concurrent thread may start from the same or different initial solutions. Here, different starting points are chosen, since this contributes to a more thorough exploration of the solution space.

3.1.3. Distribution of communication with regard to time. The concurrent threads may communicate during the search or only at the end. Strategies of the first type, often called co-operative multi-thread strategies, pursue goals such as a more thorough exploration of the solution space or a speed-up of the computation time. Strategies of the second type, also known as independent search methods, aim merely at the determination of the overall best solution. The parallel method proposed here incorporates a co-operative multi-thread strategy. The primary aim is a more thorough exploration of the solution space.

3.1.4. Synchronization of communication. Communications may be performed synchronously or asynchronously. In order to avoid waiting times of the concurrently executed processes, an asynchronously communication concept was chosen.

3.1.5. Moments of communication. Communication may be triggered by certain events, i.e. be event-driven, or executed at predetermined or dynamically decided moments. The concept of predetermined moments of communication was chosen here, since it facilitates the control of the communication in particular with respect to the frequency of communication. This is an important issue, since it affects the behaviour of the overall search and especially the balance between exploration and exploitation of (parts of) the search space. With decreasing frequency the search strategy approximates the strategy of independent search methods. A higher frequency, on the other hand, favours the intensification of the search within a certain area of the solution space.

3.2. Implementation aspects

In the following, the presentation of the implementation aspects is organized around a structural implementation view of the subject. This means that the parallelized two-phase hybrid metaheuristic is now considered as parallel-distributed system. An overview of the components of a parallel-distributed system is given in figure 2.

According to figure 2, a parallel-distributed system comprises assignment and interaction components (cf. Schütz, 1997). The assignment components concern the subdivision of the total task into subtasks, the definition of processes which concurrently carry out the subtasks, and the distribution of the processes over the computing resources. The interaction components serve the control of the process execution and the cooperation of the processes. Here the assignment components are chosen as follows. The total task, i.e. the solution to a problem instance of the VRPTW, is not subdivided into subtasks. Rather, a number of *np* differently configured processes is defined, each running on a LAN-workstation and solving a complete problem instance of the VRPTW by means of a modified two-phase hybrid

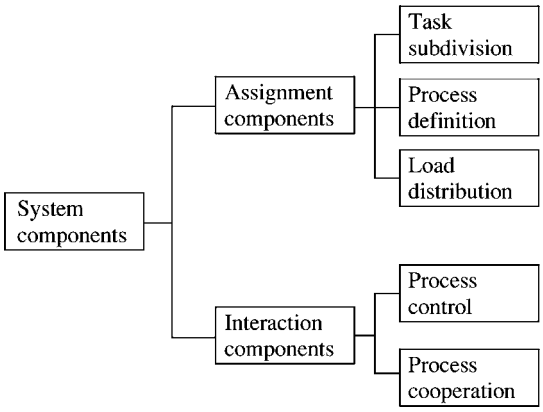

Figure 2. Components of a parallel-distributed system.

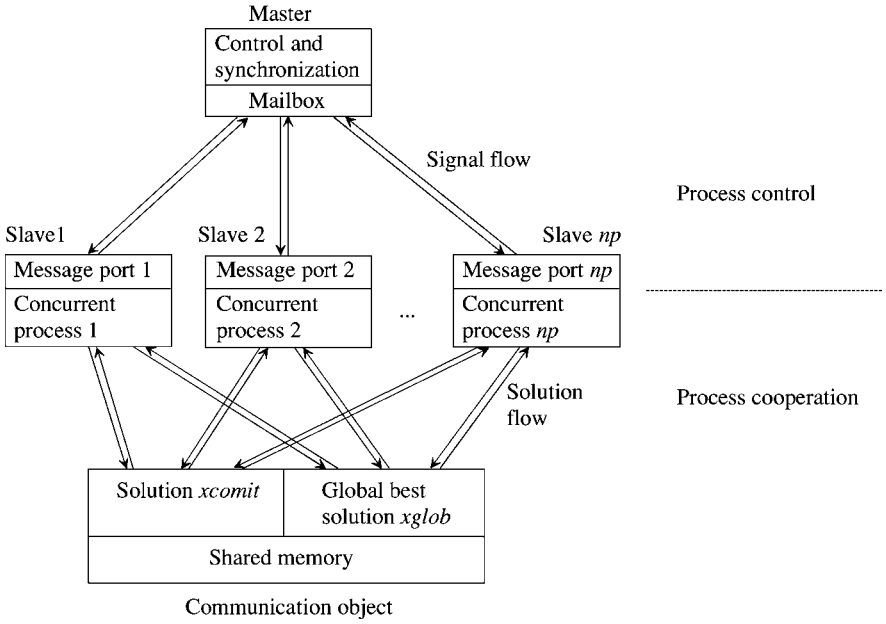

Figure 3. Architectural view of process control and process cooperation.

metaheuristic. The two-phase hybrid metaheuristic has to be modified in order to enable a process control and a process cooperation. As to the interaction components, the process control is based on a master-slave model, and the process cooperation on a weak concept of cooperative autonomy (for the concept of cooperative autonomy see Enslow, 1978).

An architectural view of the resulting process control and process cooperation is given in figure 3. In the following, both interaction components are described in more detail.

Process control concerns the control of the process interaction and the synchronization of the cooperating processes. In the master-slave model, a master process acts as superordinate control instance and sends control signals to the concurrently executed autonomous, but subordinate, processes, the so-called slave processes. The slaves do not act as control instances. They merely receive control signals from the master, carry out the modified two-phase metaheuristic concurrently and send reply signals back to the master. The exchange of control and reply signals is supported by simple communication objects. While the slaves receive the control signals sent out by the master in so-called message ports, the reply signals they emit are collected in the master's mailbox. The mailbox and the message ports are buffering objects, each able to keep several signals.

3.2.1. Process control. The interaction between the master process and the slave processes covers three consecutive periods of time. These periods concern the creation of a communication structure, the concurrent generation of a solution to a problem instance and the dissolution of the communication structure. In the following, the control activities carried out in the three periods are outlined. The focus, however, is on the second period.

During the first period the communication structure shown in figure 3 is set up. The master creates its mailbox and the communication objects, whereas a predefined number *np* of slave processes create their message ports. The master and the slaves also generate identifiers and access paths for the mailbox, the communication object and the message ports, and enter these data into a special file within the LAN. The file entries enable the master to identify activated message ports and to include the corresponding slaves in the communication structure. On the other hand, these slaves are enabled to identify the mailbox of the master and to communicate with the master.

In the second period a problem instance is solved concurrently by the slaves included in the communication structure. The master initiates the respective calculations and coordinates the termination of the calculations. The control activities of the master and the reactions of the slaves are shown graphically in figure 4 by means of an interaction diagram. The left-hand part of the diagram describes the master process and the right-hand part one of the *np* slave processes.

According to figure 4, the interaction between the master and the slaves comprises four interaction steps. Each step is represented by a so-called interaction point. The interaction steps bring about a mutual synchronization of the activities of the master and the slaves.

In the first interaction step the master initiates the concurrent calculation of a problem instance by submitting the control signal "SIG-Run" to each of the activated *np* slaves. The master then enters a waiting cycle, i.e. the master tries repeatedly to read reply signals out of its mailbox. Having received the control signal "SIG-Run", the slaves carry out the modified two-phase metaheuristic. Each slave generates an initial population, executes the first search phase and then the second search phase. The extensions included in the second search phase concern further interaction steps.

In the second interaction step each slave sends the reply signal "SIG-Reply-Run" back to the master. This signal indicates that the termination criterion for the second search phase, i.e. the tabu search, is met. Each slave then continues to carry out iterations of the tabu search method and to read its message port after each iteration. The master remains in the waiting cycle until all slaves have responded. If this is the case, the master leaves the waiting cycle and initiates the third interaction step.

The aim of the third interaction step is to achieve a defined global system state, i.e. a concurrent and correct termination of the concurrently executed slave processes. For this purpose the master submits the control signal "SIG-End" to each of the slaves. The master then enters a waiting cycle. Since the slaves access their message ports after each iteration of the tabu search method, they will read out this signal nearly at the same time.

In the fourth interaction step the slaves react on the signal "SIG-End". Each slave submits the reply signal "SIG-Reply-End" to the master and enters a waiting cycle. As soon as the master has received this reply signal from all slaves, the global system state that is aimed at is achieved. All slaves have now finished their calculations and the exchange of solutions between slave processes is therefore finished, too. This means that the global best solution *xglob* kept in the shared memory is clearly defined and cannot be influenced further by exchanges of solutions which are not yet finished. After receiving a reply from all the slaves, the master leaves the waiting cycle and enters the third period.

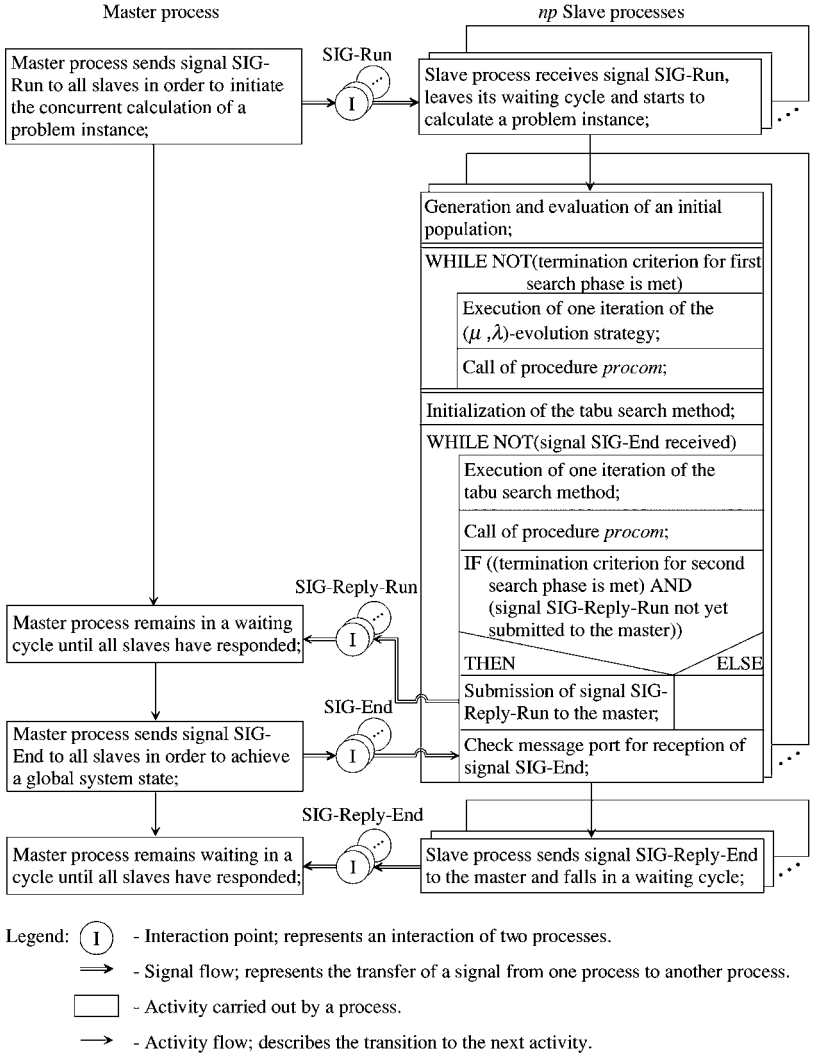

Figure 4. Interaction diagram for the concurrent calculation of a problem instance.

The third period comprises the output of the global best solution *xglob* and the dissolution of the communication structure. In order to initiate the output of *xglob*, the master submits a specific control signal to a randomly selected slave. This signal causes the slave to read *xglob* out of the communication object and to write *xglob* to an output device. The final dissolution of the communication structure is also controlled by the master. Each of the activated slave processes is caused to delete its message port and then the master deletes its mailbox.

3.2.2. Process cooperation. The process cooperation follows a weak concept of cooperative autonomy. It is characterized by four properties: (1) concurrent execution of completely

autonomous processes, (2) process cooperation through the exchange of solutions, (3) hierarchical process control according to the master-slave model, and (4) process communication by means of a shared memory. As to the process communication, the respective communication object comprises two data items within a common data area (see figure 3). The first item keeps a solution *xcomit*, which serves the exchange of solutions between the concurrently executed slave processes and the second item keeps the global best solution *xglob*, i.e. the best solution over all cooperating processes. Both data objects are from time to time accessed and updated by the cooperating slave processes.

In order to read and update the two data items, the cooperating slave-processes call the communication procedure *procom* specified in figure 5. Each of the cooperating processes initiates a procedure call after each iteration of the (μ, λ)-evolution strategy in the first search phase and, if the second search phase is executed, after each iteration of the tabu search method (see also figure 4). A procedure call passes over six parameters to procedure *procom*: (1) the current search phase *sphase*, (2) the current number of iterations *iter* carried out so far in the first search phase or in the second search phase respectively, (3) the best solution *xit* found in the *iter*-th iteration of the (μ, λ)-evolution strategy or tabu search method respectively, (4) the local best solution *xbest*, (5) the communication interval during the first search phase *cycle1*, and (6) the communication interval during the second search phase

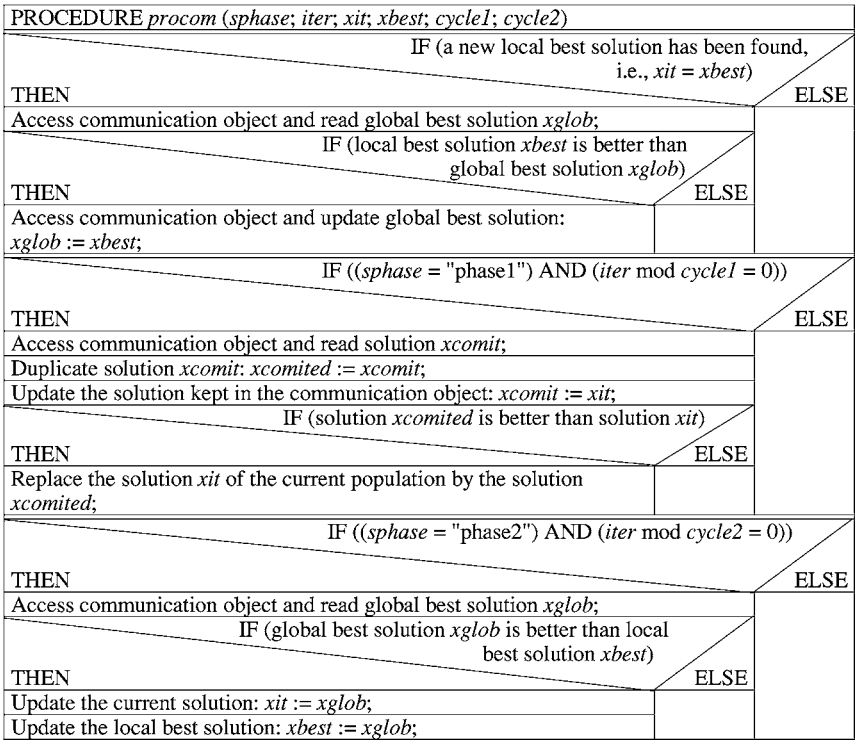

Figure 5. Structure chart for the process communication procedure.

cycle2. The values of parameter *sphase* are "phase1" for the first and "phase2" for the second search phase. In the first search phase *xit* denotes the best of the μ selected descendants in terms of the lexicographic evaluation functions introduced in Section 2. Depending on the configuration of a slave process (see Section 4), the criterion "minimal delay" or "caused overload" is used along with the other three introduced criteria. In the second search phase, however, *xit* is the best of the $c \cdot n$ generated neighbours with respect to the objective function. The communication procedure consists of three subsequent blocks. While the first block is executed in any case, the second block is carried out if the calling procedure executes the first search phase, and the third block if the second search phase is executed.

The first block serves the update of the global best solution *xglob*. This is always the case if the local best solution *xbest* of the calling process is better than the global best solution *xglob*—better in terms of the objective function. Since the first block is executed after each iteration of the (μ, λ)-evolution strategy and of the tabu search method of a slave process, the variable *xglob* finally contains the best calculated solution over all concurrently executed processes.

The second block is carried out repeatedly, i.e. always after a number of *cycle1* iterations of the respective (μ, λ)-evolution strategy, if the calling process executes the first search phase. The aim is to enhance the chances for finding a solution with a smaller number of vehicles. For this purpose, the solution *xcomit* kept in the shared memory is used to replace the best solution *xit* within the current population of the calling process if solution *xcomit* is better than solution *xit*. The operator "better" refers here to the number of vehicles and, in the case of equal numbers of vehicles, to the number of customers in the smallest route. If solution *xit* is replaced by *xcomit*, the calling process leaves the current search path and evolves a more promising new search path. The solution *xit* generated by the calling process is used to update the solution *xcomit* kept in the shared memory. Hence, solution *xit* is not lost if it is replaced by *xcomit*.

The third block is also carried out repeatedly, but always after a number of *cycle2* iterations of the respective tabu search method and if the calling process executes the second search phase. If the global best solution *xglob* kept in the shared memory is better than the current best solution *xbest* of the calling process in terms of the objective function, then solution *xglob* is used as the starting point for the further search of the calling process. This transition to a new search path directs the search to a more promising area of the solution space. Since the solution *xglob* is used to update the local best solution *xbest*, the same global best solution cannot be used a second time as a starting point for further searches. In other words, the exchange of solutions cannot cause cycling.

Transitions to new search paths cause an intensification of the search in certain parts of the solution space. In the first search phase, the intention is to explore wide parts of the solution space. Therefore, the fact is tolerated that the solutions which serve as starting points for an intensification of the search will often fall behind the global best solution in terms of the objective function. In the second search phase, however, the aim is to minimize the total distance for a given number of vehicles. Therefore the global best solution is always used as the starting point for an intensification of the search. The cooperating slave processes, i.e. modified two-phase hybrid metaheuristics, are configured in different ways. This is achieved by using different seed values for the generation of random numbers, more

precisely, by using seed values which are themselves randomly generated. Given the same starting point for an intensification of the search, the cooperating processes will therefore evolve different search paths.

A final comment concerns the transition from the first to the second search phase. Owing to the outlined cooperation concept, this transition does not need to be synchronized for the cooperating processes. It is rather possible that, for example, two of four cooperating processes run in the first phase and two in the second phase. The cooperation concept permits a process communication at any time and independently of the process states.

4. Evaluation of the parallelized two-phase metaheuristic

In order to demonstrate the performance of the developed parallel method, a comparative test for different problem sizes was carried out. In the following, the test problems used, the configuration of the developed method and the numerical results obtained are described.

The test covers seven groups of problem instances. These groups differ in the number of customers per instance. The first group consists of the 56 problem instances introduced by Solomon (1987). Each instance of this group comprises 100 customers. The second group comprises the two instances D417 and E417 given by Russell (1995), which are both taken from practice. Both instances comprise 417 customers. The problem instances of the residual 5 groups G02, G04, G06, G08, and G10 were generated by the authors. The size of these instances ranges from 200 up to 1000 customers and enables a test which also is based on realistically sized problem instances. For comparative tests these instances are available in the Internet (see http://www.fernuni-hagen.de/WINF/touren/menuefrm/probinst.htm).

The 56 problem instances of the first group are subdivided into 6 problem classes (C1, C2, R1, R2, RC1, RC2) with specific properties. A similar subdivision is chosen for the generated 5 problem groups. Each of these groups consists of 60 problem instances which comprise 200, 400, 600, 800, and 1000 customers per instance of the third, fourth, fifth, sixth, and seventh group. The 60 problem instances of each of these 5 groups are always subdivided into 6 problem classes which correspond to the Solomon classes. While the number of problem instances per class varies between 8 and 12 for the 56 Solomon problems, each of the other classes consists of 10 instances.

For all problem instances the location of the customers and the depot is given in a Cartesian coordinate system. The location coordinates are integer values. In accordance with the procedure usually found in the literature, the Euclidean distances d_{ij} and the travel times d'_{ij} were calculated exactly to two decimal places. To verify the feasibility and accuracy, the final results were recalculated with the greatest degree of accuracy fixed by the computer.

A multitude of approaches for solving the VRPTW has been proposed in the literature. Here, only some of the best methods published in the recent past can be considered. The methods included in the comparative test are:

1. The sequential metaheuristic from Chiang and Russell (1997).
2. The sequential metaheuristic from Liu and Shen (1999).
3. The sequential evolutionary metaheuristic from Homberger and Gehring (1999).

4. The type 3 parallelization approach from Schulze and Fahle (1999).
5. The type 3 parallelization approach from Gambardella, Taillard, and Agazzi (1999).
6. The type 3 parallelization approach from Gehring and Homberger (1999).
7. The parallelized two-phase hybrid metaheuristic as introduced here but without communication during the search, designated HM4.
8. The parallelized two-phase hybrid metaheuristic as introduced here, designated HM4C.

Each of the parallelized metaheuristics, HM4 and HM4C, includes a number of $np = 4$ slave processes. Each slave process was carried out on one of four equally configured LAN-workstations. The master process was executed on one of the workstations as well.

Method HM4 was derived from method HM4C by means of a simple modification: the elimination of the second and the third block of the process communication procedure *procom*, i.e. only the first block of the procedure *procom* was used. The result is that in the case of method HM4 four search paths are evolved independently of each other and the best calculated solution is determined as the solution to the problem. A comparison of the solutions obtained by the methods HM4 and HM4C will reveal the synergetic effect caused by the process cooperation.

The configuration of the methods HM4 and HM4C concerns the population size (μ), the size of the neighbourhood in the first and the second search phase (λ and $c \cdot n$), the length of the tabu list (*tabul*), the number of iterations in the first and the second search phase (*maxit1*, *maxit2*, and *stopit*), the lexicographic function *eval* for evaluating a solution xn' in the first search phase, and the number of iterations defining the communication frequency in the first and the second search phase (*cycle1* and *cycle2*). Some of these configuration parameters are constant, some vary with the problem size, and some vary between the four slave processes which constitute the parallel methods HM4 and HM4C. The values of the constant parameters are: $\mu = 30$, $\lambda = 200$, $tabul = 10$, $eval =$ "mdel" for the first three slaves, $eval =$ "ovld" for the fourth slave, and $cycle1 = 3$. The values of the remaining parameters are given in Table 1.

This parameter setting was determined in two steps. In the first step the values of the parameters $c \cdot n$, μ, λ, *tabul*, *stopit*, *cycle1*, and *cycle2* were fixed as given above or in Table 1 respectively. For the remaining parameters *maxit1* and *maxit2* higher values, as shown in Table 1, were chosen. In the second step three series of multiple experiments including two randomly selected instances from each class of problems and about 500 calculation runs were carried out. The subject was an experimental test of the values of the parameters μ and λ (first series), *tabul* (second series), and *maxit1* and *maxit2* (third series). The test design and the results can be summarized as follows:

– The population size μ was varied within the interval [1, 100] and the number of descendants λ within the interval [1, 1000]. Independent of the absolute values of μ and λ, "good" solutions were obtained for a value of $\mu/\lambda \approx 1/7$ for the quotient μ/λ. Therefore, the values $\mu = 30$ and $\lambda = 200$ fixed in the first step remained unchanged.
– The length *tabul* of the tabu list *Tlist* was varied within the interval [5, 25]. A significant impact on the solution quality was not observed. Therefore the predetermined value $tabul = 10$ remained unchanged as well.

Table 1. Configuration of the method HM4C.

			Problem groups				
Parameters	Solomon	Russell	G02	G04	G06	G08	G10
$c \cdot n$	$20 \cdot 100 = 2000$	$20 \cdot 417 = 8320$	$20 \cdot 200 = 4000$	$20 \cdot 400 = 8000$	$20 \cdot 600 = 12000$	$20 \cdot 800 = 16000$	$20 \cdot 1000 = 20000$
maxit1 (slave 1)	10000	400	200	400	600	800	1000
maxit1 (slave 2,3,4)	100000	4000	2000	4000	6000	8000	10000
maxit2	100000	4000	2000	4000	6000	8000	10000
stopit	100	400	200	400	600	800	1000
cycle2	50	200	100	200	300	400	500

– The number of iterations *maxit1* and *maxit2* were both varied within the interval [100, 1000000]. An increase in the number of iterations led, as expected, to better solutions, but also to longer computing times. Taking account of this trade-off, the values given in Table 1 were finally chosen.

As to the parameter *maxit1*, a small value is always chosen for one of the concurrent four processes. Since the respective process enters the second search phase earlier than the other processes, the chances of finding a solution with a smaller total travel distance are enhanced. Further, only one process considers the overload criterion, while the other processes use the delay criterion. The reason is that a reduction of the number of vehicles may be prevented more often by time window constraints than by capacity constraints.

The significance of the results of a comparative test may suffer both from different test conditions and from different objective functions. Test conditions of high impact are the number of runs for deriving a result and the underlying computer platform. Apart from the number of vehicles and the total travel distance some authors use the total travel time as an alternative or additional optimization criterion. The methods considered here use throughout a lexicographic objective function comprising two or three criteria. Details of the test conditions and the objective function are given in Table 2. The numbers for the optimization criteria indicate priorities.

Since optimal solutions are known for some problem instances only, the methods are evaluated on the basis of best solutions. In the following, the achieved results are presented as averages over each class of problems. The mean number of vehicles *MNV*, the mean total travel distance *MTD*, and the mean computing time *MCT* are always calculated. The calculation of these mean values is always based on the best solutions that were achieved for the problem instances of the respective problem class. The reported computing times are wall-clock times. Furthermore, the cumulated number of vehicles *CNV* and the cumulated total travel distance *CTD* are determined for each problem group.

The results obtained for the 56 problem instances from Solomon are shown in Table 3. They may be summarized as follows:

– The results achieved by the parallel method HM4C are comparable to those generated by the best sequential and parallel methods. It should be mentioned, however, that each run of the parallel method from Gehring and Homberger (1999) stops after 300 seconds.

Table 2. Test conditions and optimization criteria for the considered methods.

	Number of runs				Optimization criteria		
Method	Solomon problems	Russell problems	Groups G02–G10	Computer	Number of vehicles	Total travel distance	Total travel time
Chiang and Russell (1997)	4	–	–	PC Pentium, 166 MHz, 71 Mflops, 233 Mips	1	2	3
Liu and Shen (1999)	18	27	–	HP-9000/720, 17 Mflops, 55 Mips	1	2	–
Homberger and Gehring (1999)	10	5	–	PC Pentium, 200 MHz, 107 Mflops, 349 Mips	1	2	–
Schulze and Fahle (1999)	Multiple runs	–	–	Motorola 604, 133 MHz, 200 Mflops	1	–	2
Gambardella, Taillard, and Agazzi (1999)	Multiple runs	–	–	Sun UltraSparc 1, 167 MHz, 70 Mflops	1	–	2
Gehring and Homberger (1999)	1	–	1	PC Pentium, 200 MHz, 107 Mflops, 349 Mips	1	2	–
Parallelized metaheuristic HM4	5	5	3	PC Pentium, 400 MHz, 239 Mflops, 940 Mips	1	2	–
Parallelized metaheuristic HM4C	5	5	3	PC Pentium, 400 MHz, 239 Mflops, 940 Mips	1	2	–

– The comparison of the results achieved by the methods HM4 and HM4C indicates a small synergetic effect.

In the case of the two problems from Russell (1995) the number of vehicles *NV*, the total travel distance *TD*, and the computing time *CT* are reported instead of mean values (see Table 4). Table 4 contains the best values derived in a varying number of runs. Most of the methods generate the minimum numbers of vehicles. The significantly shorter travel distances calculated by the parallel methods HM4 and HM4C are partly due to longer computing times. On the other hand, the parallel method HM4C achieves the results from Liu and Shen (1999) after just 416 seconds for D417 and 112 seconds for E417.

The results, i.e. averages *MNV*, *MTD*, and *MCT* for the last five problem groups with a problem size ranging from 200 up to 1000 customers, are given in Table 5. (For the purpose

Table 3. Comparison of results for the problem instances from Solomon (1987).

		Sequential methods			Parallel methods				
Problem class	Mean values	Chiang and Russell (1997)	Liu and Shen (1999)	Homberger and Gehring (1999)	Schulze and Fahle (1999)	Gambardella, Taillard, and Agazzi (1999)	Gehring and Homberger (1999)	HM4	HM4C
C1	*MNV*	**10.00**	**10.00**	**10.00**	**10.00**	**10.00**	**10.00**	**10.00**	**10.00**
	MTD	828.38	830.06	828.38	828.94	828.38	829	828.50	828.63
	MCT	644	1320	522	–	–	300	94.33	103
C2	*MNV*	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**	**3.00**
	MTD	591.42	591.03	589.86	589.93	589.86	590	589.88	590.33
	MCT	1441	215	780	–	–	300	91	99
R1	*MNV*	12.17	12.17	**11.92**	12.25	12.00	12.41	12.08	12.00
	MTD	1204.19	1249.57	1228.06	1239.15	1217.73	1201	1208.14	1217.57
	MCT	5395	2657	750	–	–	300	1212	1304
R2	*MNV*	**2.73**	2.82	**2.73**	2.82	**2.73**	2.91	**2.73**	**2.73**
	MTD	986.32	1016.58	969.95	1066.68	967.75	945	965.46	961.29
	MCT	6115	399	960	–	–	300	1000	1204
RC1	*MNV*	11.88	11.88	11.63	11.75	11.63	12.00	**11.50**	**11.50**
	MTD	1397.44	1412.87	1392.57	1409.26	1382.42	1356	1389.65	1395.13
	MCT	2842	1828	660	–	–	300	957	1088
RC2	*MNV*	**3.25**	**3.25**	**3.25**	3.38	**3.25**	**3.25**	**3.25**	**3.25**
	MTD	1229.54	1204.87	1144.43	1286.05	1129.19	1140	1126.22	1139.37
	MCT	3866	427	990	–	–	300	1324	1420
All	*CNV*	411	412	**406**	413	407	417	407	**406**
	CTD	58502	59317	57876	60346	57525	56956	57420	57641

Table 4. Comparison of results for the problem instances from Russell (1995).

		Sequential methods			Parallel methods	
Problem instance	Values	Chiang and Russell (1997)	Liu and Shen (1999)	Homberger and Gehring (1999)	HM4	HM4C
D417	*NV*	55	**54**	**54**	**54**	**54**
	TD	3455.28	3747.52	4703	3498.67	3512.01
	CT	2280	2843	3523	7915	8402
E417	*NV*	55	**54**	**54**	**54**	**54**
	TD	3796.61	4691.14	4732	3836.38	3832.24
	CT	2160	2475	3492	8088	8655

Table 5. Comparison of results for the problem instances of groups G02, G04, G06, G08, and G10.

		Problem group G02			Problem group G04			Problem group G06			Problem group G08			Problem group G10		
Problem class	Mean values	Gehring and Homberger (1999)	HM4	HM4C	Gehring and Homberger (1999)	HM4	HM4C	Gehring and Homberger (1999)	HM4	HM4C	Gehring and Homberger (1999)	HM4	HM4C	Gehring and Homberger (1999)	HM4	HM4C
C1	*MNV*	**18.90**	**18.90**	**18.90**	**38.00**	**38.00**	**38.00**	57.90	57.90	**57.70**	76.70	76.60	**76.10**	96.00	96.00	**95.40**
	MTD	2782	2838.93	2842.08	7584	7853.09	7855.82	14792	14802.68	14817.25	26528	26500.48	26936.68	43273	43409.82	43392.59
	MCT	600	152	252	1200	313	556	1800	552	925	2400	923	1675	3000	1365	2533
C2	*MNV*	**6.00**	**6.00**	**6.00**	**12.00**	**12.00**	**12.00**	17.90	17.90	**17.80**	24.00	23.90	**23.70**	30.20	29.80	**29.70**
	MTD	1846	1852.63	1856.99	3935	3969.36	3940.19	7787	7758.30	7889.96	12451	11947.43	11847.92	17570	17408.71	17574.72
	MCT	600	75	84	1200	696	911	1800	1335	1719	2400	2585	3201	3000	3901	4004
R1	*MNV*	**18.20**	**18.20**	**18.20**	**36.40**	**36.40**	**36.40**	**54.50**	**54.50**	**54.50**	**72.80**	**72.80**	**72.80**	**91.90**	**91.90**	**91.90**
	MTD	3705	3808.27	3855.03	8925	9396.99	9478.22	20854	21441.67	21864.47	34586	34646.69	34653.88	57186	56445.13	58069.61
	MCT	600	85	105	1200	217	240	1800	509	513	2400	746	783	3000	1445	1077
R2	*MNV*	**4.00**	**4.00**	**4.00**	**8.00**	**8.00**	**8.00**	**11.00**	**11.00**	**11.00**	**15.00**	**15.00**	**15.00**	**19.00**	**19.00**	**19.00**
	MTD	3055	3095.33	3032.49	6502	6603.38	6650.28	13335	13520.26	13656.15	21697	21741.75	21672.85	31930	32004.02	31873.62
	MCT	600	72	82	1200	161	165	1800	341	330	2400	554	529	3000	794	794
RC1	*MNV*	**18.00**	**18.00**	18.10	**36.10**	**36.10**	**36.10**	55.10	55.10	**55.00**	72.40	**72.30**	**72.30**	**90.00**	90.10	90.10
	MTD	3555	3717.96	3674.91	8763	9139.82	9294.99	18411	18930.50	19114.02	38509	39740.69	40532.35	50668	50904.62	50950.14
	MCT	600	84	86	1200	224	338	1800	462	474	2400	1019	1107	3000	1099	1106
RC2	*MNV*	**4.30**	4.40	4.40	**8.60**	8.90	8.80	**11.80**	12.10	11.90	**16.10**	**16.10**	**16.10**	19.00	18.70	**18.50**
	MTD	2675	2651.35	2671.34	5518	5614.49	5629.43	11522	11597.51	11670.29	17741	17837.64	17941.23	27012	27039.33	27175.98
	MCT	600	87	134	1200	242	328	1800	524	699	2400	788	1043	3000	1371	1314
All	*CNV*	**694**	695	696	**1390**	1393	1392	2082	2085	**2079**	2770	2767	**2760**	3461	3455	**3446**
	CTD	176180	179645	179328	412270	425771	428489	867010	880509	890121	1515120	1524147	1535849	2276390	2272116	2290367

of comparisons, the best solutions generated in multiple experiments for these problem instances are shown in the Appendix.) The results may be summarized as follows:

- For problem groups G02 and G04 the cumulated number of vehicles *CNV* is slightly higher for the parallel method HM4C than for the method from Gehring and Homberger (1999), but significantly smaller for the remaining groups G06, G08, and G10. The reduction of the cumulated number of vehicles increases with the problem size.
- The cumulated total travel distance *CTD* for the parallel method HM4C is, for all problem groups, higher than for the method from Gehring and Homberger (1999). The reason is that the computing times for the latter method are significantly higher than for method HM4C.
- A synergetic effect cannot be observed for problem group G02. For the remaining groups, however, a synergetic effect increasing with the problem size can be observed. In the case of group G10, the parallelization causes a reduction of the number of vehicles by 1 unit for 9 of 60 problem instances.

To sum up the results that were obtained it can be stated that the developed parallel method HM4C is undoubtedly on a par with other sequential and parallel methods for solving smaller instances of the VRPTW. For more realistically sized problems with up to 1000 customers HM4C also seems to be an appropriate solution approach. On the other hand, the observed synergetic effect is limited only and raises the question whether the parallelization effort is justified. In this context two parallel aspects are of interest: the achieved speed-up and the interaction between the number of processors and the solution quality.

According to Toulouse, Cranic, and Gendreau (1996) statements on the speed-up of a type 3 parallelization approach are not meaningful, since the respective sequential and parallel metaheuristics evolve different search paths. It appears appropriate instead to compare the computing times which are necessary for achieving a given solution quality. Following this idea, two additional calculations runs were carried out for each instance of the problem groups G02 to G10. The first run served the calculation of a solution by means of the sequential two-phase metaheuristic. In the second run the computing time was determined which the parallel method HM4C required in order to derive a solution of at least the same quality. The parallel method HM4C achieved the same or even better solution quality in about 40% of the computing time required by the sequential two-phase metaheuristic.

An investigation of the interaction between the number of processors and the solution quality should cover computing times as well. In some additional experiments the impact of the number of processors on the total travel distance and on the computing time required for achieving a given total travel distance was determined. As the scaling base the number *np* of slave processes involved in the parallel metaheuristic was used and varied according to the number of processors, i.e., one slave process was always executed on each processor. Typical results for an instance of problem group C1_2_2 are shown in figure 6.

An increase in the number of processors causes first a significant decrease of the total travel distance (see figure 6(a)). The inclusion of more than five processors leads to lower improvements. The computing time required for achieving a certain total travel distance shows similar behaviour (see figure 6(b)). Comparable results were derived for other problem

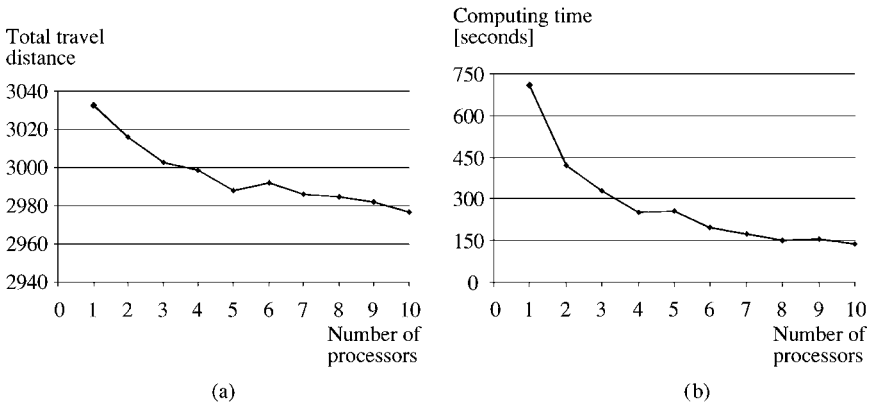

Figure 6. Interaction between the processor number and the solution quality. (a) Development of the total travel distance; (b) Development of the computing time.

instances. The determination of the number of processors or slave processes respectively within the range of 4 to 6 seems therefore to be a justifiable design decision.

5. Conclusions

This paper presents a parallel metaheuristic for solving the vehicle routing problem with time windows (VRPTW). The approach follows a type 3 parallelization strategy (cf. Toulouse Cranic, and Gendreau, 1996), i.e. several concurrent searches of the solution space are carried out with a differently configured metaheuristic. The autonomous processes cooperate through the exchange of solutions. The proposed parallel metaheuristic was subjected to a comparative test including well-known benchmark problems and new problem instances with up to 1000 customers as well.

The numerical results obtained show that the developed parallel metaheuristic is undoubtedly on a par with other methods for solving smaller instances of the VRPTW. Numerical results from other authors are not yet available for more realistically sized problems with up to 1000 customers. The solutions to these problems generated with the presented method and the required computing times seem, however, to be of reasonable quality. On the other hand, the applied parallelization concept causes a small synergetic effect only and the question arises whether this concept is justified. Some additional experiments revealed considerable gains in terms of the computing time required to generate solutions of a given quality. The type 3 parallelization strategy appears therefore to be a suitable concept for solving larger instances of the VRPTW.

Appendix

The best results, i.e. number of vehicles *NV* and total travel distance *TD*, obtained in multiple experiments for the problem instances of the problem groups G02, G04, G06, G08, and G10 are shown in the Table A.

Table A. Best results obtained in multiple experiments for the problem instances of the problem group G02, G04, G06, G08, and G10.

Problem group G02			Problem group G04			Problem group G06			Problem group G08			Problem group G10		
Problem instance	*NV*	*TD*	Problem instance	*NV*	*TD*	Problem instance	*NV*	*TD*	Problem instance	*NV*	*TD*	Problem instance	*NV*	*TD*
C1_2_1	20	2704.57	C1_4_1	40	7152.06	C1_6_1	60	14095.64	C1_8_1	80	25184.38	C110_1	100	42478.95
C1_2_2	18	2972.24	C1_4_2	37	7517.45	C1_6_2	56	14332.05	C1_8_2	75	25518.17	C110_2	93	44459.70
C1_2_3	18	2788.77	C1_4_3	36	8035.08	C1_6_3	56	14581.08	C1_8_3	72	27341.54	C110_3	90	45798.40
C1_2_4	18	2721.86	C1_4_4	36	7524.01	C1_6_4	56	14381.27	C1_8_4	72	27120.46	C110_4	90	44218.87
C1_2_5	20	2702.05	C1_4_5	40	7152.06	C1_6_5	60	14104.35	C1_8_5	80	25167.67	C110_5	100	42549.13
C1_2_6	20	2701.04	C1_4_6	40	7153.46	C1_6_6	60	14093.83	C1_8_6	80	25167.00	C110_6	100	42479.15
C1_2_7	20	2701.04	C1_4_7	39	8043.18	C1_6_7	59	14659.74	C1_8_7	79	25518.85	C110_7	99	42711.39
C1_2_8	19	2850.54	C1_4_8	38	7282.85	C1_6_8	57	14976.88	C1_8_8	76	25809.11	C110_8	96	43311.84
C1_2_9	18	2737.34	C1_4_9	36	8200.65	C1_6_9	56	15151.15	C1_8_9	73	26451.62	C110_9	91	45386.93
C1_210	18	2699.21	C1_410	36	8459.12	C1_610	56	14886.40	C1_810	72	29536.81	C11010	90	47277.00
Mean	18.90	2757.86	Mean	37.80	7651.99	Mean	57.60	14526.24	Mean	75.90	26281.56	Mean	94.90	44067.14
C2_2_1	6	1931.44	C2_4_1	12	4116.14	C2_6_1	18	7776.16	C2_8_1	24	11677.72	C210_1	30	16900.40
C2_2_2	6	1863.16	C2_4_2	12	3934.73	C2_6_2	18	7512.35	C2_8_2	24	11449.65	C210_2	29	18312.76
C2_2_3	6	1777.57	C2_4_3	12	3844.14	C2_6_3	17	8371.07	C2_8_3	23	12374.07	C210_3	29	17807.78
C2_2_4	6	1742.00	C2_4_4	12	3831.58	C2_6_4	17	7668.05	C2_8_4	23	11938.45	C210_4	29	17447.09
C2_2_5	6	1879.01	C2_4_5	12	3941.05	C2_6_5	18	7578.67	C2_8_5	24	11454.85	C210_5	30	16586.46
C2_2_6	6	1857.75	C2_4_6	12	3877.65	C2_6_6	18	7500.99	C2_8_6	24	11418.93	C210_6	30	16473.14
C2_2_7	6	1849.46	C2_4_7	12	3910.61	C2_6_7	18	7761.43	C2_8_7	24	11791.88	C210_7	31	17662.34
C2_2_8	6	1824.43	C2_4_8	12	3803.17	C2_6_8	18	7483.18	C2_8_8	24	11413.35	C210_8	29	18662.10
C2_2_9	6	1834.00	C2_4_9	12	3924.39	C2_6_9	18	7380.21	C2_8_9	24	11736.54	C210_9	30	16984.60
C2_210	6	1811.39	C2_410	12	3729.84	C2_610	17	8024.43	C2_810	23	12236.32	C21010	29	16555.24
Mean	6.00	1837.02	Mean	12.00	3891.33	Mean	17.70	7705.65	Mean	23.70	11749.18	Mean	29.60	17339.19

R1_2_1	20	4829.21	R1_4_1	40	10709.18	R1_6_1	59	21875.04	R1_8_1	80	37666.58	R110_1	100	54724.64
R1_2_2	18	4089.57	R1_4_2	36	9400.75	R1_6_2	54	21720.35	R1_8_2	72	35474.00	R110_2	91	56367.45
R1_2_3	18	3446.20	R1_4_3	36	8210.84	R1_6_3	54	19031.71	R1_8_3	72	31693.32	R110_3	91	50610.36
R1_2_4	18	3115.52	R1_4_4	36	7633.72	R1_6_4	54	17202.58	R1_8_4	72	29716.77	R110_4	91	47060.67
R1_2_5	18	4208.96	R1_4_5	36	10139.75	R1_6_5	54	25901.77	R1_8_5	72	38633.08	R110_5	91	70838.01
R1_2_6	18	3659.90	R1_4_6	36	8881.72	R1_6_6	54	20291.10	R1_8_6	72	33388.80	R110_6	91	54952.03
R1_2_7	18	3175.75	R1_4_7	36	8067.97	R1_6_7	54	18570.23	R1_8_7	72	30737.79	R110_7	91	50040.50
R1_2_8	18	2999.35	R1_4_8	36	7548.05	R1_6_8	54	17030.55	R1_8_8	72	29282.40	R110_8	91	46554.80
R1_2_9	18	3889.96	R1_4_9	36	9347.85	R1_6_9	54	23250.86	R1_8_9	72	34976.41	R110_9	91	61862.40
R1_210	18	3349.06	R1_410	36	8585.78	R1_610	54	21385.42	R1_810	72	33964.51	R11010	91	59813.39
Mean	18.20	3676.35	Mean	36.40	8852.56	Mean	54.50	20625.96	Mean	72.80	33553.37	Mean	91.90	55282.43
R2_2_1	4	4502.17	R2_4_1	8	9403.89	R2_6_1	11	18833.86	R2_8_1	15	29383.35	R210_1	19	43975.28
R2_2_2	4	3703.86	R2_4_2	8	7811.05	R2_6_2	11	15412.26	R2_8_2	15	23942.54	R210_2	19	35901.41
R2_2_3	4	2972.04	R2_4_3	8	6230.11	R2_6_3	11	12216.49	R2_8_3	15	19146.48	R210_3	19	27165.30
R2_2_4	4	2034.24	R2_4_4	8	4515.45	R2_6_4	11	8896.13	R2_8_4	15	14423.95	R210_4	19	20049.26
R2_2_5	4	3415.44	R2_4_5	8	7416.89	R2_6_5	11	15861.48	R2_8_5	15	25645.41	R210_5	19	37681.39
R2_2_6	4	2959.14	R2_4_6	8	6383.54	R2_6_6	11	13164.53	R2_8_6	15	21720.99	R210_6	19	32167.55
R2_2_7	4	2532.95	R2_4_7	8	5378.69	R2_6_7	11	10697.64	R2_8_7	15	18030.14	R210_7	19	25339.69
R2_2_8	4	1872.11	R2_4_8	8	4349.54	R2_6_8	11	8473.53	R2_8_8	15	14030.42	R210_8	19	19431.42
R2_2_9	4	3115.60	R2_4_9	8	6711.32	R2_6_9	11	14416.87	R2_8_9	15	23797.45	R210_9	19	35197.73
R2_210	4	2745.45	R2_410	8	6094.22	R2_610	11	13105.21	R2_810	15	21977.85	R21010	19	32622.36
Mean	4.00	2985.30	Mean	8.00	6429.47	Mean	11.00	13107.80	Mean	15.00	21209.86	Mean	19.00	30953.14
RC1_2_1	18	3946.11	RC1_4_1	36	9432.62	RC1_6_1	55	19610.49	RC1_8_1	73	33213.55	RC110_1	90	51529.12
RC1_2_2	18	3468.85	RC1_4_2	36	8587.76	RC1_6_2	55	17413.64	RC1_8_2	72	39696.20	RC110_2	90	48146.09
RC1_2_3	18	3201.95	RC1_4_3	36	8032.76	RC1_6_3	55	16425.70	RC1_8_3	72	35577.87	RC110_3	90	46201.21
RC1_2_4	18	2963.00	RC1_4_4	36	7749.44	RC1_6_4	55	15847.45	RC1_8_4	72	32654.10	RC110_4	90	44072.32

(*Continued on next page.*)

Table A. (*Continued*).

Problem group G02			Problem group G04			Problem group G06			Problem group G08			Problem group G10		
Problem instance	*NV*	*TD*	Problem instance	*NV*	*TD*	Problem instance	*NV*	*TD*	Problem instance	*NV*	*TD*	Problem instance	*NV*	*TD*
RC1_2_5	18	3902.58	RC1_4_5	36	9372.21	RC1_6_5	55	19769.65	RC1_8_5	73	32860.34	RC110_5	90	52780.36
RC1_2_6	18	3632.80	RC1_4_6	36	9285.81	RC1_6_6	55	18482.75	RC1_8_6	73	32701.01	RC110_6	90	52842.71
RC1_2_7	18	3449.83	RC1_4_7	36	9069.14	RC1_6_7	55	18747.28	RC1_8_7	72	43829.43	RC110_7	90	51230.16
RC1_2_8	18	3277.21	RC1_4_8	36	8698.14	RC1_6_8	55	18188.52	RC1_8_8	72	43694.60	RC110_8	90	50648.85
RC1_2_9	18	3385.56	RC1_4_9	36	8551.29	RC1_6_9	55	17825.26	RC1_8_9	72	41816.70	RC110_9	90	50354.98
RC1_210	18	3260.31	RC1_410	36	8252.50	RC1_610	55	17823.34	RC1_810	72	41182.44	RC11010	90	49307.75
Mean	18.00	3448.82	Mean	36.00	8703.17	Mean	55.00	18013.41	Mean	72.30	37722.62	Mean	90.00	49711.36
RC2_2_1	6	3136.85	RC2_4_1	11	7019.89	RC2_6_1	15	13275.93	RC2_8_1	20	20869.21	RC210_1	22	31457.37
RC2_2_2	5	2902.23	RC2_4_2	10	5945.68	RC2_6_2	12	12071.40	RC2_8_2	17	18672.43	RC210_2	19	28071.59
RC2_2_3	4	2637.41	RC2_4_3	8	5298.85	RC2_6_3	11	10466.72	RC2_8_3	15	15711.84	RC210_3	18	22680.56
RC2_2_4	4	2084.89	RC2_4_4	8	3882.74	RC2_6_4	11	8016.95	RC2_8_4	15	12314.59	RC210_4	18	17971.20
RC2_2_5	4	3087.98	RC2_4_5	9	6292.63	RC2_6_5	13	12324.05	RC2_8_5	16	19639.76	RC210_5	18	29560.06
RC2_2_6	4	3138.02	RC2_4_6	8	6054.21	RC2_6_6	12	12250.82	RC2_8_6	15	19807.15	RC210_6	18	29044.12
RC2_2_7	4	2550.58	RC2_4_7	8	5727.01	RC2_6_7	11	11849.83	RC2_8_7	15	18070.18	RC210_7	18	27954.23
RC2_2_8	4	2341.34	RC2_4_8	8	5093.18	RC2_6_8	11	11108.08	RC2_8_8	15	16947.24	RC210_8	18	26158.82
RC2_2_9	4	2286.77	RC2_4_9	8	4745.35	RC2_6_9	11	10598.17	RC2_8_9	15	16811.00	RC210_9	18	25289.27
RC2_210	4	2092.74	RC2_410	8	4418.64	RC2_610	11	9927.76	RC2_810	15	15792.31	RC21010	18	24863.21
Mean	4.30	2625.88	Mean	8.60	5447.82	Mean	11.80	11188.97	Mean	15.80	17463.57	Mean	18.50	26305.04

Acknowledgment

The authors wish to thank the anonymous referees whose constructive and valuable comments have helped us to improve this paper.

References

Chiang, W.-C. and 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. and R.A. Russell. (1997). "A Reactive Tabu Search Metaheuristic for the Vehicle Routing Problem with Time Windows." *INFORMS Journal on Computing* 9(4), 417–430.

Clarke, G. and J.W. Wright. (1964). "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points." *Operations Research* 12, 568–581.

Crainic, T.G. and M. Toulouse. (1998). "Parallel Metaheuristics." In T.G. Crainic and G. Laporte (eds.), *Fleet Management and Logistics.* Norwell, MA: Kluwer Academic Publishers, pp. 205–251.

Crainic, T.G. and M. Toulouse. (2001). "Parallel Strategies for Meta-Heuristics." In F. Glover and G. Kochenberger (eds.), *State-of-the-Art Handbook in Metaheuristics.* Norwell, MA: Kluwer Academic Publishers, to appear.

Desrochers, M., J.K. Lenstra, M.W.P. Savelsbergh, and F. Soumis. (1988). "Vehicle Routing with Time Windows: Optimization and Approximation." In B.L. Golden and A.A. Assad (eds.), *Vehicle Routing: Methods and Studies*. Amsterdam: Elsevier Sciences Publishers, pp. 65–84.

Enslow, H.P. (1978). "What is a 'Distributed' Data Processing System?" *Computer* 11, 13–21.

Gambardella, L.M., E.D. Taillard, and G. Agazzi. (1999). "MACS-VRPTW: A Multiple Ant Colony System for Vehicle Routing Problems with Time Windows." Technical Report IDSIA-06-99, Istituto Dalle Molle di Studi sull'Intelligenza Artificiale, Lugano, CH.

Garcia, B.-L., J.-Y. Potvin, and J.-M. Rousseau. (1994). "A Parallel Implementation of the Tabu Search Heuristic for Vehicle Routing Problems with Time Window Constraints." *Computers & Operations Research* 21(9), 1025–1033.

Gehring, H. and J. Homberger. (1999). "A Parallel Hybrid Evolutionary Metaheuristic for the Vehicle Routing Problem with Time Windows." In K. Miettinen, M.M. Mäkelä, and J. Toivanen (eds.), *Proceedings of EUROGEN99–Short Course on Evolutionary Algorithms in Engineering and Computer Science*, pp. 57–64. Reports of the Department of Mathematical Information Technology, No. A 2/1999, University of Jyväskylä, Finland.

Gietz, M. (1994). *Computergestützte Tourenplanung mit zeitkritischen Restriktionen*. Heidelberg: Physica.

Glover, F. (1989). "Tabu Search—Part I." *ORSA Journal on Computing* 1(3), 190–206.

Glover, F. (1990). "Tabu Search—Part II." *ORSA Journal on Computing* 2(1), 4–32.

Glover, F., J.P. Kelley, and M. Laguna. (1995). "Genetic Algorithms and Tabu Search: Hybrids for Optimization." *Computers & Operations Research* 22, 111–134.

Glover, F. and M. Laguna. (1993). "Tabu Search." In C.R. Reeves (ed.), *Modern Heuristic Techniques for Combinatorial Problems*. Oxford: Blackwell Scientific Publications, pp. 70–150.

Hansen, P. and N. Mladenovic. (1999). "An Introduction to Variable Neighborhood Search." In S. Voss, S. Martello, I.H. Osman, and C. Roucairol (eds.), *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*. Norwell, MA: Kluwer Academic Publishers, pp. 433–458.

Homberger, J. and H. Gehring. (1999). "Two Evolutionary Metaheuristics for the Vehicle Routing Problem with Time Windows." In G. Laporte and F. Semet (eds.), *Metaheuristics for Location and Routing Problems. Information Systems and Operational Research* 37(3) (special issue), 297–318.

Lenstra, J.K. and A.H.G. Rinnooy Kan. (1981). "Complexity of Vehicle Routing and Scheduling Problems." *Networks* 11, 221–227.

Liu, F.-H. and S.-Y. Shen. (1999). "A Route-Neighborhood-Based Metaheuristic for Vehicle Routing Problem with Time Windows." *European Journal of Operational Research* 118, 485–504.

Or, I. (1976). "Traveling Salesman-Type Combinatorial Problems and their Relation to the Logistics of Blood Banking." Ph.D. thesis, Department of Industrial Engineering and Management Science, Northwestern University, Evanston, IL.

Osman, I.H. (1993). "Metastrategy Simulated Annealing and Tabu Search Algorithms for the Vehicle Routing Problem." *Annals of Operations Research* 41, 421–451.

Potvin, J.-Y. and S. Bengio. (1996). "The Vehicle Routing Problem with Time Windows—Part II: Genetic Search." *INFORMS Journal on Computing* 8, 165–172.

Potvin, J.-Y., T. Kervahut, B.-L. Garcia, and J.-M. Rousseau. (1996). "The Vehicle Routing Problem with Time Windows—Part I: Tabu Search." *INFORMS Journal on Computing* 8(2), 158–164.

Potvin, J.-Y. and J.M. Rousseau. (1995). "An Exchange Heuristic for Routing Problems with Time Windows." *Journal of the Operational Research Society* 46, 1433–1446.

Retzko, R. (1995). *Flexible Tourenplanung mit selbstorganisierenden Neuronalen Netzen*. Göttingen: Unitext.

Russell, R.A. (1995). "Hybrid Heuristics for the Vehicle Routing Problem with Time Windows." *Transportation Science* 29(2), 156–166.

Schulze, J. and T. Fahle. (1999). "A Parallel Algorithm for the Vehicle Routing Problem with Time Window Constraints." In J.E. Beasley and Y.M. Sharaiha (eds.), *Combinatorial Optimization: Recent Advances in Theory and Praxis, Annals of Operations Research* 86 (special issue), pp. 585–607.

Schütz, G. (1997). *Verteilt-parallele Ansätze zur Distributionsplanung.* Wiesbaden: Gabler.

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, and J.R. Schaffer. (1988). "Vehicle Routing and Scheduling Problems with Time Window Constraints: Efficient Implementations of Solution Improvement Procedures." In B.L. Golden and A.A. Assad (eds.), *Vehicle Routing: Methods and Studies*. Amsterdam: Elsevier Science Publishers, pp. 85–105.

Solomon, M.M. and J. Desrosiers. (1988). "Time Window Constrained Routing and Scheduling Problems." *Transportation Science* 22, 1–13.

Taillard, E.D. (1993). "Parallel Iterative Search Methods for Vehicle Routing Problems." *Networks* 23, 661–673.

Taillard, E.D., P. Badeau, M. Gendreau, F. Guertin, and J.-Y. Potvin. (1996). "A Tabu Search Heuristic for the Vehicle Routing Problem with Soft Time Windows." Technical report CRT-95-66, Centre de recherche sur les transports, Université de Montréal, Montréal, Canada.

Thangiah, S.R., K.E. Nygard, and P.L. Juell. (1991). "GIDEON: A Genetic Algorithm System for Vehicle Routing with Time Windows." In *Proceedings of the 7th Conference on Artificial Intelligence for Applications*. Miami, FL: IEEE Press, pp. 322–328.

Thangiah, S.R., I.H. Osman, and T. Sun. (1995). "Hybrid Genetic Algorithms, Simulated Annealing and Tabu Search Methods for Vehicle Routing Problems with Time Windows." Technical Report UKC/OR94/4, Institute of Mathematics & Statistics, University of Kent, Canterbury, UK.

Toulouse, M., T.G. Cranic, and M. Gendreau. (1996). "Issues in Designing Parallel and Distributed Search Algorithms for Discrete Optimization Problems." Publication CRT-96-36, Centre de recherche sur les transports, Université de Montréal, Montréal, Canada.