A Parallel Hybrid Evolutionary Metaheuristic for the Vehicle Routing Problem with Time Windows

Hermann Gehring and Jörg Homberger
University of Hagen, Profilstr. 8, D-58084 Hagen, Germany

Abstract:

The vehicle routing problem with time windows (VRPTW) is an extension of the well-known vehicle routing problem with a central depot. The objective function of the VRPTW considered here combines the minimization of the number of vehicles (primary criterion) and the total travel distance (secondary criterion). In this paper, a two-phase procedural approach for solving the VRPTW is parallelized. The aim of the first phase is the minimization of the number of vehicles by means of a (1, λ)-evolution strategy, whereas in the second phase the total distance is minimized using a tabu search algorithm. The parallelization of this sequential hybrid procedure follows the concept of cooperative autonomy, i.e., several autonomous sequential solution procedures cooperate through the exchange of solutions. However, exchanges of solutions lead to the corresponding jumps in the solution space only if certain acceptance conditions are met. The good performance of both the sequential and the parallel approach is demonstrated by means of well-known and new benchmark problems.

Key words:

Evolution strategy, metaheuristic, tabu search, hybridization, parallelization, vehicle routing, time windows.

1 Introduction and problem formulation

The subject of this article is the vehicle routing problem with time windows (VRPTW). The VRPTW is an extension of the well-known vehicle routing problem (VRP): customers are to be serviced from a central depot within given service time windows. The objective function considered here combines the minimization of the number of vehicles (primary criterion) and the total travel distance (secondary criterion).

The VRPTW belongs to the class of the NP-hard combinatorial optimization problems (cf. [8]). Hence, primarily heuristic procedures are suggested for solving larger problem instances of the VRPTW. Recently, quite good results have been achieved with metaheuristics such as simulated annealing (see [2]), tabu search (see e.g. [16], [3], [9]) and evolution strategies (cf. [7]). Solving the VRPTW by means of hybrid and parallel approaches has not yet been reported in the literature.

In the following, a hybrid metaheuristic consisting of two procedural phases is proposed. While the first phase serves the minimization of the number of vehicles by means of a (1,λ)-evolution strategy, the aim of the second phase is the minimization of the total travel distance using a tabu search algorithm. This two-phase procedural approach is parallelized according to a concept of cooperative autonomy. The intention underlying the parallelization is to favour the generation of better solutions – especially for large problem instances. The sequential and the parallel metaheuristics are subjected to a comparative test using benchmark problems of varying size.

2 Hybrid evolutionary metaheuristic

The proposed hybrid evolutionary metaheuristic combines a $(1,\lambda)$-evolution strategy (see [14]) and a subsequently executed tabu search. Only during the first search phase, when the $(1,\lambda)$-evolution strategy is applied, attempts are made from time to time to reduce the total vehicle number. The best solution found in the first phase, which always comprises a minimum number of vehicles, is passed over to the tabu search. No problems arise with the encoding of solutions, since the $(1,\lambda)$-evolution strategy and the tabu search both use an identical encoding: solutions to the problem are represented by sequence vectors indicating the order in which the customers are served. In the second search phase, no attempts are made to reduce the vehicle number further. The applied tabu search algorithm aims rather at a minimization of the total travel distance. The overall two-phase search process is roughly outlined in Figure 1.

```
Calculate an actual solution sact using a modified savings algorithm;
Initialize the best solution: sbest := sact;
WHILE (time limit TL1 for first search phase is not exceeded)
    WHILE (λ feasible neighbours of sact have not yet been generated)
        Select at random a move operator from the set MoveSet;
        Generate a feasible neighbour sneigh of sact using the selected move operator either once
        or, if the stagnation criterion is met, several times;
        Generate a solution sneigh' with an eventually reduced vehicle number by applying the
        modified Or-opt operator to sneigh;
        Evaluate sneigh' and insert it into the set of actual neighbourhood solutions;
    Select the best of the λ generated feasible neighbours: sbestneigh;
    Update the actual solution: sact := sbestneigh;
    If sact is better than sbest, then update best solution: sbest := sact;
Initialize tabu list TList;
WHILE (time limit TL2 for second search phase is not exceeded)
    Select at random a move operator from the set MoveSet;
    Generate a number of c·n neighbours of sact by means of the selected move operator;
    Evaluate each feasible neighbour and insert it into the set of actual neighbourhood solutions;
    Select the best of the generated feasible neighbours which is not tabu and which considers
    the aspiration criterion: sbestneigh;
    Update the actual solution: sact := sbestneigh;
    Update the tabu list Tlist;
    If sact is better than sbest, then update best solution: sbest := sact;
```

Figure 1. Structure chart for the hybrid evolutionary metaheuristic.

Initially, a start solution is calculated by means of a modified savings algorithm and used as the first actual solution in the first search phase. The modification of the savings algorithm from CLARKE and WRIGHT (cf. [4]) concerns the consideration of the imposed time window constraints.

For the generation of neighbours of an actual solution the following operators are available in the set *MoveSet*: the Or-opt move based on an exchange concept of OR (cf. [10]), the 1-interchange move introduced by OSMAN (cf. [11]) and the 2-Opt* move proposed by POTVIN and ROUSSEAU (cf. [12]). In the first search phase, the selected move operator is applied repeatedly if a stagnation criterion is met. This criterion demands that more than a given number of neighbours of actual solutions have been generated subsequently without achieving an improvement of the current best solution. The number of moves carried out repeatedly is determined at random from the interval [3,10]. For each generated feasible neighbour, an attempt is made to reduce the total number of vehicles by means of a modified Or-opt operator. This operator always tries to eliminate the smallest route of a solution (for details see [7]). The best solution is selected from the λ, $\lambda > 1$, generated feasible neighbours of an actual solution. The selection is based on four evaluation criteria which are applied in lexicographic order as follows: total number of vehicles, number of customers in the smallest route, minimal delay, total travel distance. 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 [7]). The criterion "number of customers in the smallest route" pursues the same purpose.

However, the determination of the best of the generated actual solutions *sbest* which is passed over to the second search phase considers only two evaluation criteria: the total 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 (cf. [6]) is applied. At the beginning, the tabu list *TList* is initialized as empty list. An entry is always inserted into the tabu list after the selection of the best evaluated neighbour of an actual solution. An entry consists of those connections between customers or customers and the depot which are eliminated when the transition from an actual solution to its best evaluated neighbour is performed (see [13]). The best evaluated neighbour is selected from a number of $c \cdot n$ neighbours, where c, $c > 0$, is a constant and n is the number of customers to be served. The evaluation of each of the $c \cdot n$ neighbours is based on the total number of vehicles (primary criterion) and the total travel distance (secondary criterion). However, a neighbour is only selected if none of the connections between customers or customers and the depot of the respective route schedule are set tabu, or if, otherwise, the aspiration criterion is met. According to this criterion, the tabu status of one or more connections of a route schedule is ignored if the respective neighbour represents a new best solution.

3 Parallelization of the hybrid evolutionary metaheuristic

A parallel solution approach may be motivated by, e.g., generating a result in a shorter time (speed up), solving a larger problem in a given time (scale up), calculating a better solution in a given time, improving convergence behaviour, etc. As already mentioned, the aim pursued here is to enable the generation of reasonably good solutions for larger problem instances.

The parallelization concept used 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 level is chosen. The total task, i.e., the solution of a VRPTW, is not subdivided into subtasks. In addition, several autonomous processes are defined, each running on a LAN-workstation and solving a complete problem instance of the VRPTW. The interaction between the parallel executed processes includes two areas: process control and process cooperation. Here, process control is based on a master-slave-model, and process cooperation on the concept of cooperative autonomy already mentioned.

Process control is performed on two control levels: the first level concerns control of the process interaction, and the second level the synchronization of the cooperating processes, especially with respect to the coordination of concurrent resource requirements. According to the master-slave-model, a master(process) acts as superordinate control instance and sends control signals to the parallel executed autonomous, but subordinate, processes, the so-called slave(processes). The slaves receive these signals in their message ports and perform the required actions, above all executing the outlined hybrid evolutionary metaheuristic and exchanging solutions. The slaves commit the end of these actions by sending back defined control signals to the master.

Process cooperation follows a weak concept of cooperative autonomy (for the concept of cooperative autonomy see [5]). It is characterized by four properties: concurrent execution of completely autonomous processes, process cooperation through the exchange of solutions, hierarchical process control according to the master-slave-model and use of a shared memory for the purpose of process communication. For the exchange of solutions, a communication structure based on a blackboard is introduced. The blackboard serves as common data area consisting of two stacks. Each of the cooperating slave processes writes into the common area and reads from the common area. A write operation always inserts a solution as a new top element into one of the two stacks, and a read operation always reads the current top element out of one of the two stacks. One of the two stacks is strictly associated with the first search phase of the cooperating slave processes and the other is coupled with both search phases:

- The first stack keeps a best actual solution named *sbestact*, which is – from time to time – accessed by the cooperating processes during the execution of their first search phase.
- The second stack keeps the global best solution, i.e., the best solution over all cooperating processes, designated *sglobbest*. This variable is occasionally accessed by the cooperating processes during the execution of their first and their second search phase.

In order to access the two stacks, the cooperating procedures call the communication procedure *proccom* specified in Figure 2. A procedure call takes place after each iteration in phase 1 and after each iteration in phase 2. In phase 1, as well as in phase 2, an iteration comprises the generation of a best neighbour *sbestneigh* for a given solution. A procedure call passes over four parameters to procedure *proccom*: the actual search phase *sphase* ("phase 1" or "phase 2"), a current number of iterations *iter*, the actual solution *sact* and the local best solution *sbest*. Depending on the executed phase of the calling process, either the upper or the lower part of procedure process is carried out (see Figure 2).

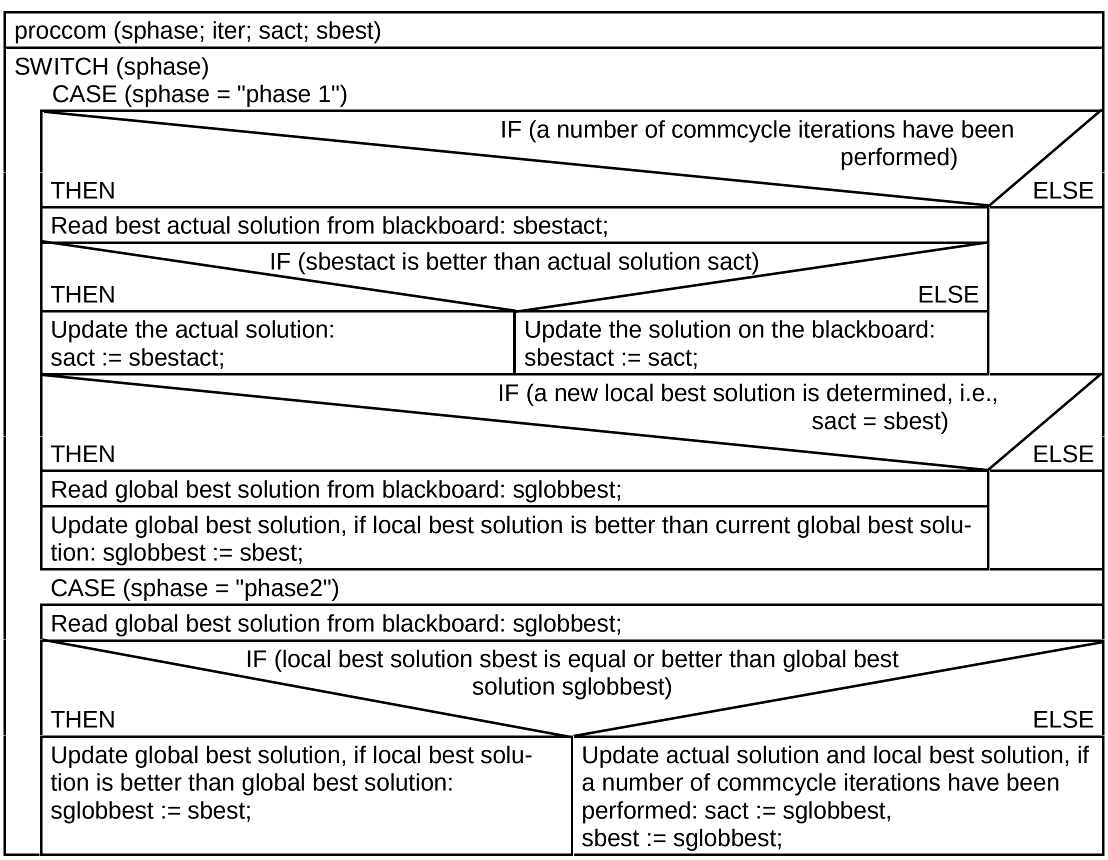

Figure 2. Structure chart for the process communication procedure.

If the upper part is carried out, i.e., the first search phase of a calling process is executed, two actions are initiated:

- If a number of *commcycle* best neighbour solutions have been generated, then the calling process eventually leaves its actual search path and continues the search with the solution kept in the first stack. This type of jump in the search space is only performed if it directs the search to a more promising new search path. Otherwise, the actual search path is continued and the actual solution inserted into the first stack.
- The global best solution is updated if the calling process has generated a new local best solution which is better than the current global best solution.

Similar actions are initiated in the lower case, i.e., the execution of the second search phase of a calling process:

- If a number of *commcycle* best neighbour solutions have been derived, then the calling process eventually continues the search with the global best solution kept in the second stack. This jump takes place if a further search in the neighbourhood of the global best solution is more promising than continuing the search with the local best solution of the calling process.
- The global best solution is updated under the same conditions as described above.

The jumps mentioned cause an intensification of the search in certain parts of the search space. In the first search phase, the intention is to explore wide parts of the search space. Therefore, solutions which fall behind the actual global solutions also serve as starting points for an intensification of the search. This concept enhances the chances of finding solutions with a minimum vehicle number. In the second search phase, however, the aim is to minimize the total distance for a given minimum vehicle number. It is therefore appropriate always to use the current global best solution as the starting point for an intensification of the search.

The cooperating slave processes, i.e. hybrid metaheuristics, are configurated 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 generated at random. Given the same starting point for an intensification of the search, the cooperating processes will therefore evolve different search paths. A repeated reading access to an unchanged top element of a stack is therefore tolerated.

A final comment concerns the transition from the first to the second search phase. Owing to the outlined cooperation concept, this transition must not 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 smoothing transition of the process states permits a process communication at any time.

4 Evaluation of the sequential and parallel methods

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

The test covers six 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 (cf. [15]). The problem instances of the residual 5 groups have been generated by the authors. For comparative tests these instances are available in the OR-Library from BEASLEY (cf. [1]) (see: http://www.fernuni-hagen.de/WINF/touren/menuefrm/probinst.htm).

Each instance of the first group comprises 100 customers. The 56 problem instances are subdivided into 6 problem classes (C1, C2, R1, R2, RC1, RC2) with specific properties. A similar subdivision is chosen for the remaining 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 second, third, fourth, fifth and sixth 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 remaining classes consists of 10 instances.

Since numerical results from other authors are not available for 5 problem groups, only three methods are included in the comparison:

- The sequential hybrid evolutionary metaheuristic (designated ES).
- A method consisting of four parallely executed but not communicating hybrid evolutionary metaheuristics (designated ES4).
- The parallel solution approach described in Section 3 consisting of four parallely executed and communicating hybrid evolutionary metaheuristics (designated ES4C).

In the case of method ES4, four solutions are generated independently for a given problem instance. The best of these four results is used as solution to the problem instance. A comparison of the results obtained by the methods ES4 and ES4C will reveal the synergetic effects caused by the process cooperation.

Apart from the seed values used for the generation of random numbers, the configuration of the hybrid metaheuristics incorporated in ES, ES4 and ES4C is identical. The configuration concerns the size of the neighbourhood in the first and second search phase (λ and $c \cdot n$), the length of the tabu List *TList* (*lTList*), the number of iterations defining the communication frequency (*commcycle*) and, finally, the termination criteria for the first and second search phase (*TL1* and *TL2*). Some of these configuration elements vary with the problem size. Details are given in Table 1.

Table 1. Configuration of the methods ES, ES4 and ES4C.

Problem size n	Neighbourhood size first phase λ	Neighbourhood size second phase c·n	Length of tabu list lTList	Communication frequency commcycle	Time limit first phase [min] TL1	Time limit second phase [min] TL2
100	5	20· 100 = 2000	10	200	1	4
200	5	20· 200 = 4000	10	180	2	8
400	5	20· 400 = 8000	10	140	4	16
600	5	20· 600 = 12000	10	100	6	24
800	5	20· 800 = 16000	10	60	8	32
1000	5	20·1000 = 20000	10	20	10	40

All calculations were carried out on a PC or a PC-LAN (Pentium processor, 200 MHz). ES, ES4 and ES4C were implemented in C and compiled with Borland 5.02.

Each of the 56 problem instances from SOLOMON and the 300 instances of the residual 5 groups was calculated once only with methods ES, ES4 and ES4C. The achieved results are presented on the level of averages over the problem classes C1, C2, R1, R2, RC1 and RC2 (see Table 2). According to the optimization criteria, the mean number of vehicles *MNV* and the mean total travel distance *MTD* are presented for each class.

In the literature, results have only been reported for the 56 problem instances from SOLOMON. A comparative test including several methods from other authors has recently been carried out by HOMBERGER and GEHRING (see [7]). The best known solutions derived from this comparative test are, approximately, obtained also with the method ES. The sequential hybrid metaheuristic seems therefore to be on a par with other methods.

Table 2. Comparison of results for six groups of problems.

Problem class	Mean values	56 SOLOMON problems (100 customers)			Second problem group (200 customers)			Third problem group (400 customers)		
		ES	ES4	ES4C	ES	ES4	ES4C	ES	ES4	ES4C
C1	MNV	**10.00**	**10.00**	**10.00**	**18.90**	**18.90**	**18.90**	38.40	38.10	**38.00**
	MTD	830	829	829	2836	2790	2782	7813	7669	7584
C2	MNV	**3.00**	**3.00**	**3.00**	**6.00**	**6.00**	**6.00**	**12.00**	**12.00**	**12.00**
	MTD	592	590	590	1913	1851	1846	3988	3939	3935
R1	MNV	**12.41**	**12.41**	**12.41**	**18.20**	**18.20**	**18.20**	**36.30**	**36.30**	**36.30**
	MTD	1203	1198	1201	3734	3710	3705	8997	8931	8925
R2	MNV	2.91	**2.82**	2.91	4.10	4.10	**4.00**	**8.00**	**8.00**	**8.00**
	MTD	955	947	945	3072	3022	3055	6618	6502	6502
RC1	MNV	12.25	**11.88**	12.00	18.10	**18.00**	**18.00**	36.30	36.20	**36.10**
	MTD	1357	1356	1356	3616	3511	3555	8939	8766	8763
RC2	MNV	**3.25**	**3.25**	**3.25**	4.50	**4.30**	**4.30**	8.80	**8.60**	**8.60**
	MTD	1154	1144	1140	2681	2658	2675	5594	5507	5518

Problem class	Mean values	Fourth problem group (600 customers)			Fifth problem group (800 customers)			Sixth problem group (1000 customers)		
		ES	ES4	ES4C	ES	ES4	ES4C	ES	ES4	ES4C
C1	MNV	58.40	58.10	**57.90**	76.90	**76.60**	76.70	97.30	96.80	**96.00**
	MTD	14914	14732	14792	26786	26641	26528	43596	43409	43273
C2	MNV	18.10	18.00	**17.90**	24.30	24.30	**24.00**	30.60	30.60	**30.20**
	MTD	7919	7807	7787	12543	12358	12451	17864	17562	17570
R1	MNV	55.40	**54.50**	**54.50**	**72.80**	**72.80**	**72.80**	92.80	**91.90**	**91.90**
	MTD	20979	20815	20854	34857	34389	34586	57911	57324	57186
R2	MNV	**11.00**	**11.00**	**11.00**	**15.00**	**15.00**	**15.00**	**19.00**	**19.00**	**19.00**
	MTD	13623	13344	13335	21770	21461	21697	32416	31971	31930
RC1	MNV	55.20	**55.10**	**55.10**	73.00	72.70	**72.40**	90.50	90.30	**90.00**
	MTD	18750	18485	18411	38873	38117	38509	51493	50734	50668
RC2	MNV	12.00	12.00	**11.80**	16.50	16.30	**16.10**	19.50	19.20	**19.00**
	MTD	11627	11464	11522	17827	17609	17741	27126	26768	27012

The results obtained for the remaining 5 problem groups may be summarized as follows:

- Method ES4 generates better results than method ES with respect to the (total) vehicle number and the (total) travel distance. As to the number of vehicles, the improvement increases with the problem size. While for the second problem group only 3 fewer vehicles are occupied in total, the total reduction amounts to 19 vehicles for the sixth group. This effect is not surprising, since the hybrid metaheuristic is a stochastic method. Hence, the best of four trials will tend to dominate over the result of only one trial.
- The comparison of the results achieved with the methods ES4 and ES4C substantiates an evident synergetic effect which increases with the problem size. The process cooperation causes a reduction of the total number of vehicles. This reduction amounts to just one vehicle for the second group, but to 17 vehicles for the sixth group.

To sum up, it can be stated that the parallelization of evolution strategies seems to be an appropriate concept for the solution of larger instances of the VRPTW and, probably, of other combinatorial optimization problems as well.

References

[1] BEASLEY, J. E. (1990) OR-Library: Distributing test problems by electronic mail. *JORS* **41**, 1069-1072.

[2] CHIANG, W.-C. and RUSSELL, R. A. (1996) Simulated annealing metaheuristics for the vehicle routing problem with time windows. *Annals of Operations Research* **63**, 3-27.

[3] CHIANG, W.-C. and RUSSELL, R. A. (1997) A reactive tabu search metaheuristic for the vehicle routing problem with time windows. *INFORMS Journal on Computing* **9**, 417-430.

[4] CLARKE, G. and WRIGHT, J. W. (1964) Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, **12**, 568-581.

[5] ENSLOW, H. P. (1978) What is a 'Distributed' Data Processing System? *Computer* **11**, 13-21.

[6] GLOVER, F. (1989) Tabu Search - Part I. *ORSA Journal on Computing* **1**, 190-206.

[7] HOMBERGER, J. and GEHRING, H. (1999) Two Evolutionary Metaheuristics for the Vehicle Routing Problem with Time Windows. Appears in*: Special issue of INFOR on "Metaheuristics for Location and Routing Problems"*.

[8] LENSTRA, J. and RINNOOY KAN, A. (1981) Complexity of vehicle routing and scheduling problems. *Networks*, **11**, 221-227.

[9] LIU, F. and SHEN, S. (1998) A route-neighbourhood-based metaheuristic for vehicle routing problem with time windows. Working paper, National Chiao University, Hsinchu, Taiwan.

[10] 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.

[11] OSMAN, I. H. (1993) Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of Operations Research*, **41**, 421-451.

[12] POTVIN, J.-Y. and ROUSSEAU, J. M. (1995) An exchange heuristic for routing problems with time windows. *Journal of the Operational Research Society*, **46**, 1433-1446.

[13] POTVIN, J.-Y., KERVAHUT, T., GARCIA, B.-L. and ROUSSEAU, J.-M. (1996) The vehicle routing problem with time windows - part I: tabu search. *INFORMS Journal on Computing*, **8**, 158-164.

[14] RECHENBERG, I. (1973) *Evolutionsstrategie*. Fromman-Holzboog, Stuttgart.

[15] SOLOMON, M. M. (1987) Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, **35**, 254-265.

[16] TAILLARD, E., BADEAU, P., GENDREAU, M., GUERTIN, F. and POTVIN, J.-Y. (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.