TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Verkehrswissenschaften „Friedrich List“

DISKUSSIONSBEITRÄGE AUS DEM INSTITUT FÜR WIRTSCHAFT UND VERKEHR

NR. 2/2017

STEPHAN HOCKE, CHRISTINA GAJEWSKI, MATHIAS KASPER

A GENETIC ALGORITHM FOR VEHICLE ROUTING PROBLEMS WITH TEMPORAL SYNCHRONIZATION CONSTRAINTS

HERAUSGEBER: DIE PROFESSOREN DES INSTITUTS FÜR WIRTSCHAFT UND VERKEHR
ISSN 1433-626X

In den Diskussionsbeiträgen aus dem Institut für Wirtschaft und Verkehr der TU Dresden erscheinen in zeitlich loser Folge verkehrswirtschaftliche Arbeiten von allgemeinem Interesse. Die Diskussionsbeiträge enthalten Vorträge, Auszüge aus Diplomarbeiten, interessante Seminararbeiten, verkehrswirtschaftliche Thesenpapiere, Übersichtsarbeiten, ebenso wie Beiträge, die zur Veröffentlichung in referierten Zeitschriften vorgesehen sind. Allen Beiträgen gemeinsam ist wissenschaftliche Fundierung und wissenschaftlicher Anspruch, jedoch je nach Zweck des jeweiligen Beitrages in unterschiedlichem Maße. Die in diesem Diskussionsbeitrag vertretenen Standpunkte liegen ausschließlich in der Verantwortung der Autoren und decken sich nicht zwingend mit denen der Herausgeber.

Als Herausgeber fungieren die Professoren des Instituts für Wirtschaft und Verkehr der TU Dresden.

A Genetic Algorithm for Vehicle Routing Problems with Temporal Synchronization Constraints

Stephan Hocke, Christina Gajewski and Mathias Kasper

Technische Universität Dresden, Fakultät Verkehrswissenschaften "Friedrich List",
Professur für Verkehrsbetriebslehre und Logistik, 01062 Dresden, Germany
Stephan.Hocke@tu-dresden.de, Christina.Gajewski@tu-dresden.de,
Mathias.Kasper@tu-dresden.de

Abstract

This paper presents a Genetic Algorithm for the Vehicle Routing and Scheduling Problem with time windows and temporal synchronization constraints. That means that as opposed to the usual procedure, in addition to the usual task covering, some vertices must be served by more than one vehicle at the same time. The chromosome coding used here is based on a proposed solution representation by Mankowska et al. [19]. The Genetic Algorithm is able to solve their instance types up to 20 vertices near to optimality. Even in greater instances with 100 vertices the solution quality of the Genetic Algorithm outperforms the Local Search presented by Mankowska et al. [19], however with losses in runtime. In order to get more comparable results, both solution approaches are evaluated at the well-known benchmark instances of Bredström and Rönnqvist [6]. This includes the presentation of a simple repair algorithm during the chromosome crossover based on an insertion heuristic in order to achieve the hard time window constraints of the benchmarks.

1 Motivation

The problem to create the most cost efficient routes from a depot to a set of geographically scattered customers was first mathematically described by Dantzig and Ramser in 1959 [8]. During the last 50 years the so-called Vehicle Routing Problem (VRP) has opened up a new research area in the operational research. It was subject of countless scientific publications, which can be justified by its intellectual challenge and its fundamental practical relevance in the field of transportation, distribution and logistics [15]. The research in the field of VRP and its variants is ongoing, motivated by still unsolved theoretical as well as practical problems [10].

The Vehicle Routing Problem with synchronization constraints (VRPS) is recently one of the most investigated extensions of the classical VRP. Synchronization in the context of this paper implies that routes depend on each other

in spatial and temporal aspects. That means that there is at least one vertex which requires simultaneous operations of two vehicles or service operators, respectively. Since cooperation between teams/workers is widely held precondition to accomplish a task in the real world, the problem arises in various contexts. Ioachim et al. [13] describe a weekly fleet assignment and routing problem. Because of marketing purposes uniform flights on different days have to be scheduled at the same time. The study by Dohn et al. [9] focuses on the scheduling of ground handling tasks in some of Europe's major airports. In some cases the cooperation of several teams is required to complete one task between the arrival and the subsequent departure of an aircraft.

Not only in the aviation industry teams must be formed, deployed, and disband flexibly, Li et al. [16] tackle an efficient manpower allocation and scheduling problem for the Port of Singapore. Another example are cable companies providing internet services, not infrequently services jobs require a combination of different technicians whose visits must be synchronized or must fulfill precedence constraints [24]. Salazar-Aguilar et al. [25] mentioned synchronization problems at snow plowing operations. Since some roads have multiple lanes, the plowing and spreading deicers operations on these roads have to be done by a synchronized fleet of vehicles. Amaya et al. [1,2] study the road network marking in Quebec, where service vehicles can be refilled at certain points, provided that a corresponding refill vehicle was spatially and temporally synchronized. But also the Home-Health-Care sector is a well studied application of the VRPS. Some health care services enforce simultaneous cooperation of at least two nurses at the home of one patient, e. g. for lifting of a disabled person. Furthermore, some drugs must be administered a certain time before providing a meal or medications may require monitoring at a later time of the day [19]. Another context of temporal dependencies are forest operations described by Bredström and Rönnqvist [6]. Since forwarding can only be done once the harvesting has been performed, a coordination of both teams is mandatory.

The wide spread of VRPS increases the need for efficient solutions procedures, however, the additional set of synchronization constraint forecloses an optimal solution for real world problem sizes. Even heuristic approaches struggle to find good solutions in justifiable computation time. Due to interdependencies caused by synchronization constraints the change of one route may affect other routes and their feasibility. This constitutes a fundamental difference to the traditional VRP. Consequently, many established approaches and solution procedures are not directly applicable. To the best of our knowledge, there is no Genetic Algorithm (GA) adaption for the VRPS. Due to synchronization constraints, the established chromosome decoding (splitting) process by Dynamic Programming (DP) is excluded [7,22]. It is therefore impossible to label the auxiliary graph completely, because decision stages (nodes), which require a simultaneous operation, cannot be evaluated. The aim of this paper is to overcome this problem.

The remainder of the paper is organized as follows. In section 2 we present some background information about different modeling approaches of temporal synchronization and the applied solutions methods. Section 3 provides a prob-

lem description and defines the integer problem formulation. Subsequently, we explain the adaption for our GA in section 4, which includes a modified permutation solution representation by [19] for chromosome decoding, a corresponding Append-Heuristic (AH) for the chromosome encoding, crossover procedure, population development and a repair algorithm to achieve possible hard time window constraints. The application is successfully tested on the provided problem instances of [19]. However, these results are based on a very specific problem formulation. To make the proposed procedure more comparable, the GA evaluates the benchmark (BM) instances of Bredström and Rönnqvist [6]. The results are presented in section 5 and some concluding remarks are given in section 6.

2 Literature Review

Due to the variedness of VRPS applications, literature offers a wide range of solution approaches. An essential difference of these strategies lies in the mathematical formulation of synchronization constraints. The use of a vehicle independent time variable T_i determining the operation start time at vertex i is sufficient to ensure temporal synchronization, providing an arc-variable based formulation [9,16,17]. In doing so, there is no need for further explicit constraints linking the scheduling variables of all involved vehicles [10]. The drawback is, that no precedence can be expressed. Because of this, the majority of publications use a vehicle-dependent scheduling variable T_{ik} within an arc-variable based formulation.

Regardless of the modeling these time variables, the methods of decomposition - column generation and Branch-and-Price - are considered to be more difficult. Usually, the time variable remains in the master problem in an path-variable formulation. The dual variables of the corresponding constraints induce linear costs on the vertices in the subproblems, which requires a non-trivial adaption of the standard labeling algorithms. Ioachim et al. [13] and Bélanger et al. [4] present an advanced labeling procedures to deal with this issue. Dohn et al. [9] achieve a master problem which does not contain the scheduling variable T_i. This is possible because a column within their Branch-and-Price approach introduces an additional time dimension, containing every possible start time within the schedule horizon, measured in minutes. That means, to meet a customers demand for vehicles, the master problem has to obtain a vehicle-path-combination with identical operation start times for the associated vertex of every involved vehicle or column, respectively.

Moreover, to obtain synchronization within the master problem several authors propose to branch on time windows [4,5,9,13,23]. That means, if a solution of the relaxed master problem contains paths visiting the same vertex at different times, two new branches are created. This is done by splitting a fractal T_i or in the middle of two different T_{ik} contravening the synchronization constraint. The goal is to limit the possible operation start time more and more, until ultimately only one possible start time remains. In many cases, it becomes necessary to check the feasibility of all paths within the current tableau of the master prob-

lem with regard to the new time window restrictions. However, a prerequisite is that the time assumes as a discrete dimension, which is not a requirement of reality. Furthermore, branching on time windows is not sufficient to guarantee the integerity of a solution, mostly branching on the common path-variables is used at the end [10].

Haddadene et al. [12] investigate the influence of different objectives and formulations of the VRPS as mixed integer linear program (MILP) using IBM Optimization Programming Language (OPL). Kergosien et al. [14] propose several cutting techniques and a two criteria objective in a lexicographic way to improve the exploration significantly. The first term is a cost criteria representing the accumulated distance or travel time, respectively. The second term depicts a corrective term that helps to avoid the exploration of identical solutions considering homogeneous vehicles. Bredström and Rönnqvist [6] developed a matheuristic, which iteratively solves their MILP formulation [5], wherein the amount of arcs contained in the graph is greatly reduced. In every iteration only some randomly generated additional arcs and arcs used in the current best known solution are allowed. If a better solution is obtained, it serves as the best known solution in the next iteration.

However, the synchronization constraints are not only difficult to handle for exact procedures. Most heuristics exploit the independence of routes, e. g. the (intra and inter) shift and swap operators of Local Search (LS). Note that the determination of scheduling variables and feasibility of not directly affected routes is time intensive. Even a simple reallocation of one vertex within a route may require the rescheduling of all other routes. To avoid this, Eveborn et al. [11] split all vertices which require a synchronization between two vehicles into two independent visits and fix a common starting time of those visits. They formulate the resulting Vehicle Routing Problem with time windows (VRPTW) as minimum matching problem. Subsequently, a repeated matching algorithm is used to assign the vertices and vehicles or routes, respectively. The described procedure is part of a decision support system LAPS CARE, which is currently in operation at a number of home care providers in Sweden.

There are only few publications using established LS operators. Lim et al. [17] apply shift, exchange and rearrange operator to explore the neighborhood of a feasible solution. Since they collect information about the feasibility of an arc during the graph construction, the feasibility check of these operators can be done in constant time. Unfortunately, they do not give any information about this procedure. Due to the fact that they not only apply their Local Search operators on individual vertices but entire route segments, probably a related method to Vidal et al. [33] was used, which would require a preprocessing with the time complexity $O(n^2)$. Since Vidal et al. [33] do not consider interdependencies between routes, it may be necessary to repeat this preprocessing after each move. In addition, Lim et al. [17] do not describe how their operators deal with synchronization vertices. Even with a vehicle-independent scheduling variable T_i, the arrival times of all participating vehicles at the associated vertex are required regardless of whether a synchronization customer is directly affected by

an Local Search operator or not. Since whole route segments were allocated it would be very sophisticated by a single arc validation to ensure a consistent order of customers to be synchronized across all routes. An example of an inconsistent ordering is illustrated in figure 1. This problem occurs if at least two synchronization vertices are operated by the same vehicles. In both corresponding routes the associated vertices must be served in the same order.

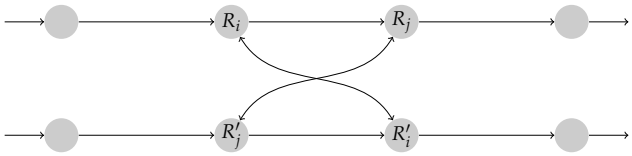

Fig. 1. Inconsistent order of customers to be synchronized across two routes

The implementation of Append Heuristics (AH) in various procedures turned out to be particularly promising. The concept is to represent the solution as a permutation of vertices. Iteratively, from the first element of the permutation to the last element, the corresponding vertex is added to the end of a specific route or two routes, respectively, if the vertex requires a simultaneous operation. Because the inserted vertex is always the current last, rescheduling is not required, which represents the main advantage of these procedures. All vertices, which require synchronization, are fully processed within one iteration. That means, that the operation start time at the associated vertex is simultaneously determined for all involved vehicles in one step. In addition, a consistent order of these customers across routes is guaranteed. Within the permutation, the traditional LS-operators can be used. However, the LS-operator usually requires a reconstruction of the solution by AH. This approach is part of different heuristic procedures involving LS [15,16,19]. Labadie et al. [15] generate a solution from a given sequence or permutation, respectively, of all customers by a straight forward algorithm. At each iteration a set of vehicles is determined which are able to serve the associated vertex. If several vehicles can fulfill the operation without time window violation, the vehicle with the lowest travel costs is selected. In case of insufficient number of vehicles to serve a customer, the solution is infeasible and is discarded. Starting from a feasible permutation Labadie et al. [15] apply a relocate operator during an iterative Local Search. Subsequently, the detailed solution is obtained by the AH.

Li et al. [16] propose a Simple-Append and a Block-Insertion heuristic to construct a solution from a given permutation. Since their objective is to minimize the total number of workers and new hires are allowed, these procedures always generate a feasible solution no matter what the input sequence is. So they can easily apply a Block-Transposition and a Block-Reverse to generate neighbors within a Simulated Annealing approach. Mankowska et al. [19] developed a matrix solution representation including all vehicle-vertex assignments in a column

wise notation. They introduce eight different Local Search operators which can be applied to their solution representation. At each iteration of their AH the vertex within one column is scheduled at the end of the routes of all vehicles that have an entry of the associated vertex in their row. Furthermore, they use a soft constraint for the latest possible operation start time of a vertex. In doing so, Mankowska et al. [19] obviate time consuming checks of time window compliance. Moreover, this ensures that their operators produce feasible solutions exclusively. Overall, they introduce three different LS procedure, which includes a steepest descent search, a merged neighborhood search and an adaptive variable neighborhood search. All procedures were successfully tested at real world instances with up to 300 vertices.

A further solution approach used is constraint programming (CP). The CP paradigm has been successful in solving hard combinatorial problems especially in tightly constrained problems. Problems are expressed in terms of variables, domains and constraints specifying which assignments of values in the domain of variables are allowed. The solution procedure uses complete search techniques such as depth-first search and branch and bound. Rousseau et al. [24] use a Constrained-Based-Insertion to generate a initial solution, which is further improved by Local Search operators and Tabu-Search. Their application was successfully tested at Solomon [30] benchmark instances.

3 Problem formulation

Let $G = (\bar{N}, A)$ be a complete directed graph, where $\bar{N} = \{o, d, 1, ..., n\}$ is a node set and $A = \{(i,j) | i \neq j, i \in \bar{N} \setminus \{d\}, j \in \bar{N} \setminus \{o\}\}$ is the arc set. For each arc $(i,j) \in A$ the weight d_{ij} is defined by the positive distance or traveling time, respectively. The nodes o and d represent the central depot for a fleet of vehicles K and the nodes $N = \{1, ..., n\}$ are the customers to be served. Due to synchronization $N = N^R \cup N^S$ contains two disjunct subsets. Let N^S denote the set of all vertices which require synchronized operations of two vehicles and let N^R denote the set of all regular vertices which can be proceeded by a single vehicle. Moreover, for each vertex $i \in N$ an associated time window is defined as $[e_i, l_i]$, where $e_i \geq 0$ and $l_i \geq 0$ terminate the earliest and the latest possible start time of a vehicle operation at vertex i. The duration of an operation at vertex $i \in N$ is equal to h_i.

We use a preprocessing step to generate the subset K_i including all compatible vehicles for a customer $i \in N$ and the subset A_k including all relevant arcs for a vehicle $k \in K$, if there is a heterogeneous fleet. In doing so, the model is able to express required vehicle qualifications for serving a customer $i \in N$ or possible preferences between vehicles and customers. The presented core model of VRPS (equations (1)-(7)) is an approach to unite the problem formulations of Bredström and Rönnqvist [6] and Mankowska et al. [19]. The model contains at least two decision variables, the binary routing variable $X_{ijk} \in \{0, 1\}$ and the scheduling variable $T_{ik} \geq 0$. Furthermore, we introduce the binary $Y_k \in \{0, 1\}$ to consider if a vehicle is used in the solution or not.

$$\sum_{k \in K_i} \sum_{j|(i,j) \in A_k} X_{ijk} = \begin{cases} 1 & \text{if } i \in N^R \\ 2 & \text{otherwise} \end{cases} \quad \forall i \in N \tag{1}$$

$$\sum_{j|(o,j) \in A_k} X_{ojk} = \sum_{j|(j,d) \in A_k} X_{jdk} = 1 \quad \forall k \in K \tag{2}$$

$$\sum_{j|(o,j) \in A_k \wedge j \in N} X_{ojk} \leq Y_k \quad \forall k \in K \tag{3}$$

$$\sum_{j|(i,j) \in A_k} X_{ijk} - \sum_{j|(j,i) \in A_k} X_{jik} = 0 \quad \forall i \in N, k \in K_i \tag{4}$$

$$T_{ik} + d_{ij} + h_i - M(1 - X_{ijk}) \leq T_{jk} \quad \forall k \in K, (i,j) \in A_k \tag{5}$$

$$M \left(2 - \sum_{j|(j,i) \in A_k} X_{jik} - \sum_{j|(j,i) \in A_l} X_{jil} \right) \geq T_{il} - T_{ik} \quad \forall i \in N^S, l, k \in K_i \tag{6}$$

$$-M \left(2 - \sum_{j|(j,i) \in A_k} X_{jik} - \sum_{j|(j,i) \in A_l} X_{jil} \right) \leq T_{il} - T_{ik} \quad \forall i \in N^S, l, k \in K_i \tag{7}$$

Constraint (1) guarantees that every vertex $i \in N$ is served according to it's requirements. Since some vertices require a simultaneously treatment, the number of incoming arcs must be equal to 1 or 2, respectively. If a vehicle $k \in K$ is used in a solution, it's origin o and destination d must be the central depot (2). In constraint set (3) the Y_k is linked to the routing variable X_{ijk}. Hence, $Y_k = 1$ if a vehicle k leaves the origin o to serve a vertex $i \in N$. Equation (4) represents the flow conservation constraint. Constraints (5) determine the start times T_{ik} of the operations with respect to the proceeding time h_i and the traveling time d_{ij}. It ensures that T_{ik} of operations along the route of a vehicle k are strictly increasing. Since a return to an already served vertex would violate the start time of the prior operations, the constraints (5) avoid cycles, so no further sub-tour elimination is necessary.

The equations (6) and (7) are the additional synchronization constraints. The constraint sets enforce the equality of start times T_{ik} and T_{il} providing vehicles k and l serve vertex i. If further temporal dependencies of operations like precedence should be considered a maximum time distance $\delta_i^{\max}$ and minimum time distance $\delta_i^{\min}$ could be added at the left hand side of the constraints (6) and (7), respectively.

The following constraints represent two different modeling approaches of time windows. Both differ in their handling of delays. These approaches are briefly presented separately, since the developed GA can be applied to both. Bredström and Rönnqvist [6] strictly forbid start times earlier than e_i and later than l_i. Their corresponding constraint (8) imposes that $T_{ik} = 0$ in case of $\sum_{j|(j,i)} X_{jik} = 0$, meaning that the hard time window constraint is binding, if and only if, the vehicle k performances the operations at vertex i directly after the operation at vertex j.

$$e_i \sum_{j|(i,j)\in A_k} X_{ijk} \leq T_{ik} \leq l_i \sum_{j|(i,j)\in A_k} X_{ijk} \quad \forall i \in N, k \in K_i \tag{8}$$

Unlike Bredström and Rönnqvist [6], Mankowska et al. [19] allow an exceedance of l_i, which is expressed by a tardiness Z_i in constraint (9). So in case of soft time window constraint l_i, there is at least one additional non-negative decision variable $Z_i \geq 0$. Furthermore, Mankowska et al. [19] introduced $Z_{max} \geq 0$ denoting the maximal tardiness observed in the whole solution in equation (10). In doing so, they avoid an unbalanced distribution of tardiness Z_i between customers $i \in N$. That means, it should be ruled out that only a few customers accumulate a large amount of tardiness at the favor of the remaining customers.

$$T_{ik} \leq l_i + Z_i \qquad \forall i \in N, k \in K_i \tag{9}$$

$$Z_{max} \geq Z_i \qquad \forall i \in N \tag{10}$$

Note, this MILP does not contain an objective. There are a several publications with many different objectives, which mostly depend on additional side constraints caused by their specific applications. A variety of objective can already be applied for the model presented here. An examination of some of the most common objective criteria is done in the computational study in section 5.

4 Genetic Algorithm

Genetic Algorithms are randomized metaheuristics inspired by natural evolution. Classical GAs maintain a population of chromosomes that encode the properties of the corresponding individuals. The objective value of each solution represents the fitness of the associated individual, e. g. distance or travel time. The reproduction process, known as crossover, is stochastically biased, since individuals who are better adapted to their environment imply a higher probability of reproduction. The crossover selects two parent individuals and combines their most promising features to create a new solution. The resulting offsprings exhibit some characteristics of each parent. Furthermore, random mutations provide the necessary diversity of the population to avoid premature convergence. According the Darwinian principle only the fittest individuals of each generation survive.

Most of the adaptations of the classical GA to the VRP differ in terms to chromosome encoding. In several publications this aspect is completely renounced to apply various operations directly on the solution [21,28]. However, due to the fact that these procedures cannot guarantee a consistent order of customers to be synchronized across all routes, sophisticated repair algorithms would be necessary to obtain feasibility in the context of VRPS [29]. An example for an inconsistent order can be seen in figure 1.

Another recurring technique is the clustering according to the polar angle or sweep-algorithm, respectively. In Thangiah [31,32] chromosomes encode a

number polar angle according to the amount of available vehicles. These angles are used to form independent clusters of customers, each served by a single vehicle. Baker and Ayechew [3] encode an ordered list of vehicle assignment (index of the associated vehicle), wherein the order is determined by the sorted polar angles of the customers. Unfortunately, this encoding is foreclose, since multiple vehicles have to operate within one cluster to obtain synchronization or customers require more than one vehicle assignment, respectively.

The last mentioned chromosome encoding is the permutation of customers without trip delimiter like in the context of Traveling Salesman Problems. In the context of the Capacitated Vehicle Routing Problem Prins [22] introduced a splitting procedure based on Dynamic Programming to ascertain the optimal solution for each permutation. This method was successfully adapted to the VRPTW by Cheng et al. [7] and to the VRP with simultaneous Pickup and Delivery and Time Windows in Liu et al. [18] and Scheffler et al. [27]. Since synchronization vertices require two assigned vehicles, the splitting by Prins [22] is also excluded. However, an AH according to Labadie et al. [15] and Li et al. [16] would be possible. Nevertheless, we have decided against it, as with this decoding algorithms, even with unlimited computation time no optimality is guaranteed. The following subsection introduces a new chromosome decoding which overcomes all problems mentioned above.

4.1 Solution representation and chromosome encoding

The developed solution representation for the chromosome encoding is based on a shortened matrix representation of Mankowska et al. [19]. The matrix $\Theta^{r\times c}$ is composed of $c = 1, \ldots, |N|$ columns and $r = 3$ rows. The top row r_0 or the horizontal header, respectively, indicates a vertex $i \in N$. The two rows r_1 and r_2 underneath contain the indices of the vehicles $k, l \in K_{r_0}$ assigned to the vertex in row r_0 of the corresponding column. Depending on simultaneous treatment requirements, each column has one or two vehicle assignments. Table 1 shows a complete translation of a sample permutation with $|N| = 7|$ and $|K| = 3$.

Table 1. Solution representation as permutation

$$\Theta^{r\times c} = \begin{array}{c|ccccccc} & c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 \\ \hline r_0 & 7 & 2 & 4 & 3 & 6 & 5 & 1 \\ r_1 & 1 & 2 & 1 & 3 & 1 & 3 & 2 \\ r_2 & 2 & - & - & - & 3 & - & - \end{array} \rightarrow \text{AH} \rightarrow \begin{array}{ll} \text{Route } k=1: & o-7-4-6-d \\ \text{Route } k=2: & o-7-2-1-d \\ \text{Route } k=3: & o-3-6-5-d \end{array}$$

Note, that this representation can be easily adapted to modified synchronization constraints. If more vehicles are necessary for service operations on a vertex at the same time, the matrix could be extended by the number of additional rows needed. By using the row index of the assigned vehicles, precedence constraints can also be described.

A permutation of all columns (vertex-vehicle assignments) is transformed to a feasible schedule using a Simple Append Heuristic successively. The corresponding vertex i of the current column is scheduled to the route end of the assigned vehicles. For every vertex j, the arrival time is determined as $a_{jk} = d_{ij} + h_i + T_{ik}$, where i denotes the predecessor of vertex j within route k. Given that the first vertex is added to an empty route, we fix $a_{ik} = \max(e_i, d_{0i})$.

As an operation can not start before the corresponding earliest possible start e_i, we set $T_{ik} = \max(a_{ik}, e_i)\ \forall i \in N^R$. In the case of synchronization requirements we determine $T_{ik} = T_{il} = \max(a_{ik}, a_{il}, e_i)\ \forall i \in N^S$. This procedure is repeated iteratively for all columns c until the last column (vertex) is processed. Since Θ encode a complete solution and neither duration nor waiting times are part of the objective, optimality is guaranteed, providing unlimited computation time of the GA. Note, due to the simplification as to the computation of the start times this procedure is still a heuristic. Our experimental studies have shown that the ability of this proceeding to minimize route duration and waiting time is poorly evolved. However, our approach to determine the operation start times as early as possible does not differ from the most common heuristics in the context of the VRPTW. This is because this method is the most promising to obtain time window compliance. Nevertheless, it is not guaranteed in any case that the start time of each operation can be terminated before the latest possible start time l_i of the associated vertex i.

In contrast to Mankowska et al. [19], the benchmark instances of Bredström and Rönnqvist [6] include hard time windows. Regarding the operation start time, an additional procedure is necessary to obtain feasibility. So we check if $T_{ik} \leq l_i\ \forall i \in N$. For violation issues, the concerned vertex is skipped and remains in the set $U \subset N$ of unscheduled vertices for the moment.

After the complete processing of the permutation or solution representation, respectively, an insertion heuristic is used to assign every unscheduled vertex $u \in U \subset N$ to one or two (if $u \in N^S$) vehicle(s). The aim is to find the best insertion position for every $u \in U$ so that the increase in the overall cost is the lowest.

4.2 Crossover and mutation

Traditional crossover procedures can be applied to the solution representation in form of column recombination. Fig. 2 shows an Ordered-Crossover (OX). After selecting two random parent chromosomes $P1$ and $P2$, two randomized crossover points ζ_1 and $\zeta_2 = \zeta_1 + \frac{1}{2}|N|$ representing column indices within $P1$ are selected. The resulting successive partial column sequence is transfered into the offspring C. For each vertex i denoted by $\theta_{0c}^{P1} | \zeta_1 \leq c \leq \zeta_2$ within the subsequence we set a flag to track that these vertex has been transmitted. Subsequently, $P2$ is proceeded column by column starting with $c = 1$. If no flag for the associated vertex j in θ_{0c}^{P2} has been set yet, the column $\theta_{\bullet c}^{P2}$ is copied at the first vacant position within the offspring C and the transmission flag for j is set. In case of an already set flag, the column is skipped. To put it simple, the remaining columns are placed in the offspring C in the same order as they appear in $P2$.

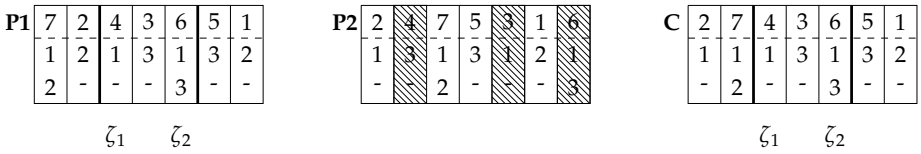

Fig. 2. Chromosome recombination with OX

Obviously, no new assignments can be created in this way. Therefore mutations are an indispensable part of our Genetic Algorithm. Since the column entries r_1 and r_2 remain unchanged during the crossover our mutation operator focuses on creating new vertex-vehicle assignments. Let i denotes the associated vertex in θ_{0c} and $k, l \in K_i$ denote the assigned vehicles in θ_{1c} and θ_{2c}. Then, the mutation operator interchange k to $k' \in K_i \setminus \{l\}$ or l to $l' \in K_i \setminus \{k\}$, respectively. During our parameter setting we achieved the best results using a mutation probability of 15 % considering the convergence of the GA, wherein 10 % of the vertex-vehicles assignments where changed. Because the detailed solution can only be generated from a complete and consecutive permutation, the mutation feasibility has not been tested. That is, any column crossover and any possible mutation of the assignment rows can be applied. The observance of the time windows is only checked within the AH.

4.3 Preprocessing and efficient feasibility testing

The efficient feasibility testing is already subject of several publications [20,26,33]. Nevertheless, these procedures need an adaption to the synchronization constraints. Before looking for the best possible insertion of the unscheduled vertices, we calculate the *Possible-Push-Forward* P_i for every already scheduled vertex i within route $R_k = \{\iota = 1, ..., |R_k|\}$, wherein $k \in K$ and ι denote the operation index within R_k. Let $\Delta_{R_{k\iota}} = l_{R_{k\iota}} - T_{R_{k\iota}}$ represent the difference of the operation start time T_i and the latest possible start time l_i of vertex i denoted by operation $R_{k\iota}$. In addition, $\nabla_{R_{k\iota}} = P_{R_{k\iota+1}} + W_{R_{k\iota+1}}$ is the sum of *Possible-Push-Forward* of the successor operation $R_{k\iota+1}$ and the waiting time $W_{R_{k\iota+1}}$ of operation $R_{k\iota+1}$. Note, since we start from a feasible solution, synchronization is already obtained, so T_i can be treated as vehicle independent variable. However, during the AH for each $R_{k\iota} \in N^S$ the corresponding partner operation $R_{l\iota'}$ must be safed. The operation pair $(R_{k\iota}, R_{l\iota'})$ represents a single vertex $i \in N^S$. The equations (11) and (12) contain the calculation rules to evaluate the *Possible-Push-Forward* which determine the maximum postponement of an operation measured in time.

$$P_{R_{k\iota}} = \min(\Delta_{R_{k\iota}}, \nabla_{R_{k\iota}}) \qquad \forall R_{k\iota} \in N^R \tag{11}$$

$$P_{R_{k\iota}} = P_{R_{l\iota'}} = \min(\Delta_{R_{k\iota}}, \nabla_{R_{k\iota}}, \nabla_{R_{l\iota'}}) \qquad \forall (R_{k\iota}, R_{l\iota'}) \in N^S \tag{12}$$

Note, the *Possible-Push-Forward* $P_{R_{k\iota}}$ for the last operation within a route is equal to $\Delta_{R_{k\iota}}$, since the P_d of the depot d is infinity. The whole preprocessing procedure is shown in figure 3. There are two partial routes R_1 and R_2 with one operation pair $(R_{13}, R_{22}) \in N^S$ whose start time need to be synchronized. We evaluate every route R_k step-by-step from their last operation up to its first. Therefore, the parameter ω_k denotes the index of the last evaluated operation within R_k. In our example we start at $\omega_1 = 4$ and $\omega_2 = 3$. Furthermore we use a recursive implementation of the algorithm. This special programming technique breaks down the problem into smaller components. The aim of each component is to evaluate the *Possible-Push-Forward* from operation R_{kw_k} up to operation $R_{k\iota}|0 < \iota \leq \omega_k$ within R_k. If this route segment contains an operation $R_{k\iota}$ which requires a synchronization with operation $R_{l\iota'}$, the route R_l must be evaluated from $R_{l\omega_l}$ up to $R_{l\iota'}$. So the algorithm is calling itself from an appropriated *Caller-Route* to proceed a segment of a *Receiver-Route*.

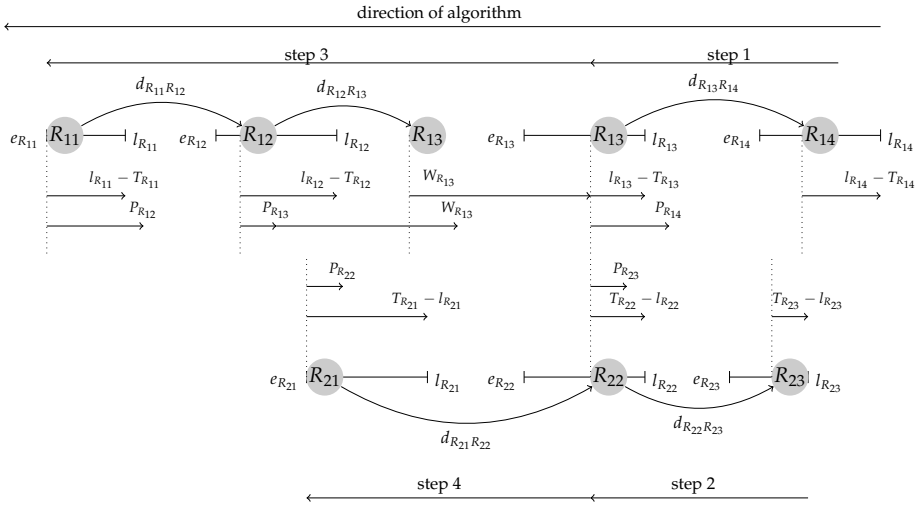

Fig. 3. Efficient preprocessing for insertion feasibility testing

In attempt to evaluate R_1 from operation R_{14} up to R_{11}, we encounter operation R_{13} (step 1) which requires a synchronization with its partner operation R_{22} within R_2. Consequently, the algorithm is calling itself to proceed route R_2 from operation R_{23} up to R_{22} (step 2). Afterwards the *Possible-Push-Forward* of R_{13} and R_{22}, respectively, can be determined according to equation (12). If the algorithm detects another operation pair $(R_{l\iota}, R_{m\iota'}) \in N^S$, which needs to be synchronized, the algorithm will call itself another time and so on. Note, since we start from a feasible solution, a consistent ordering of synchronization vertices is guaranteed, so there is no danger of much recursion (stack overflow).

That means, within one recursive call stack R_k can only be *Caller-Route* and *Receiver-Route* once, except the first *Caller-Route* which can never be *Receiver-Route* on the same call stack. After synchronization of $(R_{13}, R_{22}) \in N^S$, the original process continues from R_{13} up to R_{11} (step 3). Since $\omega_2 = 2$, in the final step we evaluate R_2 from operation R_{22} up to R_{21} (step 4).

Algorithm 1 Efficient preprocessing

```
 1: initialize: ω_k, P[k][ω_k]                          ▷ Indices and Possible-Push-Forwards
 2:
 3: proceedRegular=[&] (int k, Operation curOp) {
 4:     P[k][ω_k] ← min(Δ_curOp, ∇_curOp)
 5:     ω_k ← ω_k − 1
 6: }
 7:
 8: proceedSync=[&] (int k, int l, Operation curOp, Operation targetOp) {
 9:     P[k][ω_k] ←P[l][ω_l] ← min(Δ_curOp, ∇_curOp, ∇_targetOp)
10:     ω_k ← ω_k − 1
11:     ω_l ← ω_l − 1
12: }
13:
14: routeSyncAt=[&] (int indexReceiver, int indexCaller, Operation targetOp) {
15:     do
16:         Operation curOp ←Route[indexReceiver][ω_indexReceiver]
17:         if curOp ∈ N^R then
18:             proceedRegular(indexReceiver, curOp)
19:         else if curOp ∈ N^S and curOp ≠ targetOp then
20:             routeSyncAt(curOp.k, indexReceiver, curOp)
21:         else if curOp ∈ N^S and curOp = targetOp then
22:             proceedSync(indexCaller, indexReceiver, targetOp)
23:         endif
24:     while (curOp ≠ targetOp)
25: }
26:
27: for k ← 1 to |K| do
28:     while ω_k ≥ 1 do
29:         Operation curOp ←Route[k][ω_k];
30:         if curOp ∈ N^R then
31:             proceedRegular(k, curOp)
32:         else if curOp ∈ N^S then
33:             routeSyncAt(curOp.l, k, curOp)
34:         endif
35:     endwhile
36: endfor
```

A Pseudo-Code implementation of the preprocessing is illustrated in Algorithm 1. There are two auxiliary function *proceedRegular* (lines 3-6) and *pro-*

ceedSync (lines 8-12) which calculate the *Possible-Push-Forward* according to equations (11) and (12), respectively, and update the operation index ω_k. The used syntax is based on the definition of lambda functions within C++. Hence, the [&] implies that the whole scope is accessible within the function by reference, e.g. the *Possible-Push-Forward* array initialized in line 1.

The third lambda expression *routeSyncAt* (line 14-25) represents a recursive function which proceed a route segment within a *receiverRoute*. This function is used by a *callerRoute*, if information about the *Possible-Push-Forward* of the counterpart of $targetOp$ within another route is required. To get to this information, it is necessary to iterate from the current index ω_k over all associated operation within the *receiverRoute* till the equivalent to $targetOp$ is reached. In doing so, there are three possible cases that can occur. If the operation *curOp* is a regular vertex (line 17) the *Possible-Push-Forward* could easily be determined by calling *proceedRegular*. In the second case (line 19), *curOp* represents a vertex different from $targetOp$ which requires a synchronization. Consequently, before the *receiverRoute* can be further processed, the synchronization between the *receiverRoute* and another route must be taken into account. So, the function *routeSyncAt* is called recursively. In the third case (line 21), *curOp* is equivalent to $targetOp$, that is to say both operation correspond to the same vertex served in different routes. If so, the *Possible-Push-Forward* can be calculated according to equation (12) by calling the auxiliary function *proceedSync*. In addition, the termination criterion of the function is reached.

Finally, the nested loops in lines 27-36 describe the whole algorithm used in this paper for a preprocessing determining the *Possible-Push-Forward*. The total preprocessing has a time complexity of $O(|N| - |U| + |K|)$. Accordingly, time complexity for the evaluation of the feasibility of an insertion is reduced to $O(1)$. While inserting an unscheduled vertex $u \in U$, we maintain a queue which contains all directly and indirectly affected routes. All routes within the queue require a new preprocessing, before the next unscheduled vertex can be inserted. As soon as a permissible position for a vertex cannot be determined, the chromosome repairing fails. In this case, the solution has to be discarded.

4.4 Population development

We halve the population S for an efficient implementation of the GA. One half $SG = \{1, \ldots, |S|/2\}$ includes the fittest individuals that have been identified so far, in descending order by their fitness. The other half $SB = \{1+|S|/2, \ldots, |S|\}$ contains random solutions created by the GA so far with lower fitness than the individuals $s \in SG$ with the lowest fitness within SG ($SG_{|S|/2}$). For every crossover, we take a random element from each set. The resulting offspring s^* is mutated with a probability of 15 %. By assuming that the AH and the insertion heuristics are successful, we first check, if this fitness of s^* is better than the fitness of $SG_{|S|/2}$. If $s^* \succ SG_{|S|/2}$ and $s^* \notin SG$ we move $SG_{|S|/2}$ at a random position within SB, then s^* is sorted from behind into SG. If $s^* \prec SG_{|S|/2}$ and $s^* \notin SB$ the offspring s^* replaces a random element $s \in SB$ depending on the

fitness of s^* versus the average fitness of SB and a random factor. In both cases s^* is discarded, if $s^* \in SG$ or $s^* \in SB$, respectively.

One drawback of the proposed solution representation $\Theta^{r \times c}$ in section 4.1 is its redundancy. There are many different matrices encoding the same solution. Therefore we used the corresponding tour string with delimiter, which could be easily maintained within the AH. In case of a homogeneous fleet, this approach cannot ensure a disjunct population, however, it is still useful to avoid a too fast convergence of the GA.

Algorithm 2 Genetic Algorithm

```
 1: initialize: S from random permutations sorted by fitness
 2: initialize: SG ← {S_1, ..., S_|S|/2}
 3: initialize: SB ← {S_|S|/2+1, ..., S_|S|}
 4:
 5: while terminated = false do
 6:     Chromosome Θ* ←crossover(P1 ∈ SG, P2 ∈ SB)
 7:     mutate(Θ*, 0.15)
 8:     Solution s* ←appendHeuristic(Θ*)
 9:     if s* isFeasible then
10:         improveByLS(s*)
11:         if s* ≻ SG_|S|/2 and s* ∉ SG then
12:             insertIntoSG(s*)
13:         else if s* ≺ SG_|S|/2 and s* ∉ SB then
14:             insertIntoSB(s*)
15:         else
16:             discard(s*)
17:         endif
18:     else
19:         discard(s*)
20:     endif
21:     update(terminated)
22: endwhile
```

The whole sequence of events of the GA is displayed in Algorithm 2. At first the population sets SG and SB were initialized. Since we use the steady state approach for our GA, a more differentiated consideration of individual iterations is superfluous. The procedure is carried out until a termination criterion has been reached (line 5), e.g. a certain amount of solutions has been generated or a maximum time limit has been exceeded. In each iteration, one individual is selected from both population halves, representing the parents for the crossover described in section 2 (line 6). Subsequently, the chromosome encoding Θ^* mutates with a probability of 15 % (line 7). A detailed solution s^* determining all decision variables is created by the AH (line 8). Unless there are any unscheduled vertices $U = \varnothing$, the solution s^* is feasible. If so, s^* is insertedspe into the

existing population as described above (lines 11 - 17). Finally, the termination criteria will be updated (line 21).

Since the LS procedures developed by Mankowska et al. [19] can also be applied to the solution representation introduced in section 4.1, a hybrid implementation of the GA is also possible. If s^* is a permissible solution, LS can be used to improve the fitness before attempting to add s^* to the existing population S (line 10). However, the disadvantage of the approach is that solutions can still be discarded after a time-consuming improvement. In return, a multi threaded implementation turns out to be very simple. That means, every time a LS thread emits the signal to be finished (s^* was discarded or inserted, respectively), a connected slot of GA thread creates a new offspring s^* representing the initial solution for the restart of the LS thread.

5 Computational Results

We divide the computation results into two parts. In the main study we introduce the multi critical objectives applied to the described benchmark instances of Mankowska et al. [19] and Bredström and Rönnqvist [5]. We use GAMS 24.8/Cplex 12 running on a server with 24 threads Intel Xeon 3.47 GHz and 192 GB RAM to calculate integer solutions F or at least lower bounds LB. We limit the computing time to 10 h per instance. In doing so, we compare the determined Cplex results with the proposed GA and the proposed LS procedures of Mankowska et al. [19]. That means a Steepest Descent Search (SDS), a Merged Neighborhood Search (MN) and a Adaptive Variable Neighborhood Search (AVN). In addition, we implement a hybrid GA (hGA) which is composed of the procedures mentioned before. We tested the heuristics on a Intel Quad Core i5-4670 3.4 GHz and 12 GB RAM, furthermore the runtime is restricted to one hour. The settings are summarized in table 2.

Table 2. Computation setting

procedure	processor	threads	RAM	runtime
Cplex	Intel Xeon 3.47 GHz	24	192 GB	10 h
Heuristics	Intel i5-4670 3.4 GHz	4	12 GB	1 h

Subsequently, we evaluate the suitability of the heuristic providing different weights of the objective terms within a sensitivity analysis. In doing so, we discover a general weakness to minimize route duration of many heuristic procedures in the VRPTW context.

5.1 Main study

At first, we solve the Home-Health-Care Routing and Scheduling Problem formulated by Mankowska et al. [19]. Their modeling of a VRPS includes several

more side constraints like qualifications of vehicles which are simplified within our modeling by using vehicle and customer subsets (see section 3). However, we lose the opportunity to serve a vertex twice by one vehicle. But this is only relevant if precedence is considered. Furthermore they allow time window violations. The tardiness of a vertex operation Z_i, defined in equation (9), plus the highest observed tardiness within the whole solution Z_{max}, defined in equation (10), are included in the objective function (13).

$$\min F \rightarrow \gamma_1 \sum_{i \in N \cup \{o\}} \sum_{k \in K_i} \sum_{j|(i,j) \in A_k} d_{ij} X_{ijk} + \gamma_2 \sum_{i \in N} Z_i + \gamma_3 Z_{max} \tag{13}$$

We generated 10 instances per type (A)-(D) of the described benchmarks of Mankowska et al. [19], wherein type (A) represents 10, type (B) 25, (C) 50 and (D) 100 vertices. For the main study we applied an equal weight of the objective terms $\gamma_1 = \gamma_2 = \gamma_3 = 1/3$. The computational results can be seen in table 3. Bold column entries represent the best obtained objective value. For every instance the LS procedures solved the same 200 random initial solutions, provided there is sufficient time available. The given entries F^{SDS}, F^{MN} and F^{AVN} in table 3 correspond to the objective value of the best solution found, cpu denotes the required computational time. The same 200 random initial solutions are used as initial population of the GA and the hGA.

All algorithms find the optimal solution for type (A) within a few seconds. The GA produces on average slightly better solution quality than the LS procedures in (B) and (C). However, LS procedures require less computing time. For every instance type the developed hGA obtains the best observed objective value. In each instance, for which an optimum could be determined by Cplex, the corresponding solution was also determined by the hGA. Even in type (D) the hGA proved to be the most suitable, although the hGA procedure is only able to complete a few iterations. Nevertheless, the paradigm of evolutionary development provides more promising solutions than using LS from random initial solutions. However, using the hGA is no longer applicable for even larger problem instances with limited run time to one hour.

The solution quality of the stand-alone GA is comparable to the best solution found for all LS procedures. From a problem size of 100 vertices, the approximation time of the GA increases significantly. Nevertheless, it is noticeable that the GA, in contrast to the LS, is not polynomial. The procedure can also be carried out completely for the largest examined instances. The is not the case with any other method tested in this publication. Thus, not all 200 initial solution can be processed by the LS and the hGA shows no signs of an approximation at the time of termination.

In the second test, we solve 10 instances per type (A'), (B') and (C') according to Bredström and Rönnqvist [6], which includes 25, 50 and 80 vertices, respectively. The length of time windows for all vertices i are fixed to 120 minutes and represent hard constraints according to constraint (8).

In contrast to Bredström and Rönnqvist [6] the objective (15) used here does not contain terms for preferences and work balance of the vehicles. However, in order to achieve the goal of a balanced vehicle utilization, we tried to add the

Table 3. Mankowska et al. [19] benchmarks and objective (13)

	F	LB	cpu	F^{GA}	cpu	F^{hGA}	cpu	F^{SDS}	cpu	F^{MN}	cpu	F^{AVN}	cpu
A.0	**224.4**	224.4	<1	**224.4**	10	**224.4**	36	**224.4**	<1	**224.4**	<1	**224.4**	<1
A.1	**196.9**	196.9	<1	**196.9**	9	**196.9**	35	**196.9**	<1	**196.9**	<1	**196.9**	<1
A.2	**227.9**	227.9	<1	**227.9**	11	**227.9**	32	**227.9**	<1	**227.9**	<1	**227.9**	<1
A.3	**286.9**	286.9	<1	**286.9**	11	**286.9**	39	**286.9**	<1	**286.9**	<1	**286.9**	<1
A.4	**189.1**	189.1	<1	**189.1**	9	**189.1**	22	**189.1**	<1	**189.1**	<1	**189.1**	<1
A.5	**156.0**	156.0	<1	**156.0**	12	**156.0**	28	**156.0**	<1	**156.0**	<1	**156.0**	<1
A.6	**224.0**	224.0	<1	**224.0**	3	**224.0**	29	**224.0**	<1	**224.0**	<1	**224.0**	<1
A.7	**270.6**	270.6	<1	**270.6**	11	**270.6**	15	**270.6**	<1	**270.6**	<1	**270.6**	<1
A.8	**263.7**	263.7	<1	**263.7**	9	**263.7**	18	**263.7**	<1	**263.7**	<1	**263.7**	<1
A.9	**239.2**	239.2	<1	**239.2**	8	**239.2**	24	**239.2**	<1	**239.2**	<1	**239.2**	<1
B.0	**564.1**	564.1	6626	565.1	59	**564.1**	285	569.9	19	569.1	21	567.7	16
B.1	**326.4**	326.4	750	326.4	17	**326.4**	126	344.5	21	338.3	20	348.9	17
B.2	**421.2**	421.2	119	443.7	57	**421.2**	771	444.4	23	428.1	24	425.7	18
B.3	**482.1**	482.1	104	485.9	40	**482.1**	242	510.4	24	489.0	25	515,8	18
B.4	**432.7**	432.7	12361	**432.7**	53	**432.7**	312	449.7	18	445.2	21	450.9	16
B.5	**428.9**	428.9	423	431.3	97	**428.9**	160	438.7	22	437.8	25	451.9	18
B.6	**336.3**	336.3	3184	341.2	47	**336.3**	138	340.5	25	347.3	29	345.8	19
B.7	**459.9**	459.9	7777	**459.9**	70	**459.9**	234	474.1	22	461.9	23	472.2	17
B.8	**328.2**	328.2	1040	331.8	87	**328.2**	185	339.6	21	338.1	20	354.3	18
B.9	**424.9**	424.9	94	426.0	80	**424.9**	73	426.1	25	426.1	24	428.4	20
C.0	-	438.4	36000	725.2	385	**628.3**	1289	799.5	436	761.8	379	805.5	161
C.1	-	408.5	36000	1360.9	441	**1101.0**	2824	1432.1	673	1350.2	747	1426.7	456
C.2	1751.5	451.0	36000	612.8	534	**579.4**	1121	636.9	501	587.4	494	632.0	283
C.3	-	384.0	36000	626.4	126	**549.4**	769	666.8	439	668.1	431	669.1	263
C.4	-	397.0	36000	636.8	508	**575.4**	694	658.2	458	655.9	396	696.8	251
C.5	3099.9	443.4	36031	654.7	494	**604.2**	2657	705.9	391	683.5	422	669.8	259
C.6	-	415.9	36000	650.8	575	**548.0**	1126	608.8	229	667.5	402	651.1	159
C.7	-	397.8	36000	618.6	972	**540.0**	726	714.3	839	658.4	393	664.0	200
C.8	-	420.6	36000	661.6	970	**574.0**	1492	683.5	297	662.2	435	677.6	289
C.9	-	435.2	36000	691.3	467	**648.1**	1709	796.7	292	784.1	413	772.7	471
D.0	-	496.7	36022	1051.1	2595	**973.2**	3912	1145.5	3607	1089.0	3660	1174.2	3608
D.1	-	480.2	36016	1079.7	591	**968.2**	3760	1256.4	3618	1161.9	3663	1237.7	3606
D.2	-	489.6	36076	1126.8	2262	**993.3**	3645	1170.1	3605	1022.6	3606	1198.9	3602
D.3	-	567.6	36049	1235.2	521	**1078.0**	3849	1287.7	3642	1235.7	3627	1288.9	3606
D.4	-	510.2	36019	1127.5	1815	**1079.1**	4033	1361.6	3613	1175.2	3656	1289.9	3626
D.5	-	517.1	36020	1135.6	1849	**974.1**	3864	1165.8	3657	1094.2	3666	1135.8	3614
D.6	-	447.8	36046	1062.7	1797	**927.8**	3666	1215.4	3619	1061.5	3622	1176.7	3622
D.7	-	464.8	36017	1066.7	849	**938.3**	3965	1221.1	3667	1026.0	3626	1088.9	3615
D.8	-	483.2	36020	1028.0	1480	**929.2**	3698	1050.4	3631	1044.2	3622	1029.4	3610
D.9	-	494.9	36111	1071.7	2779	**922.3**	3656	1181.8	3614	1061.7	3637	1181.8	3628

term of route duration into the objective. Like mentioned before, the solution quality of all heuristics is weak when juxtaposed with the objective value. Since the determination of the operation start times T_{ik} focuses on time windows compliance during the AH, the ability to reduce the route duration and waiting time is poorly developed (see section 4.1). Another approach to decrease waiting time and increase utilization is to minimize the fixed costs of used vehicles within the solution. For a better comparability of the implemented procedure, we decided to limit our objective (15) to minimize the accumulated distance or travel time, respectively. That means $\lambda_1 = 1$ and $\lambda_2 = \lambda_3 = 0$ within the main study. Other lambda settings are part of the sensitivity analysis (see section 5.2).

$$L_k = T_{dk} - T_{ok} \qquad \forall k \in K \tag{14}$$

$$\min F' \rightarrow \lambda_1 \sum_{i \in N \cup \{o\}} \sum_{k \in K_i} \sum_{j|(i,j) \in A_k} d_{ij} X_{ijk} + \lambda_2 \sum_{k \in K} L_k + \lambda_3 \cdot f \cdot \sum_{k \in K} Y_k \tag{15}$$

The computational results are summarized in Table 4. Again, bold entries represent the best obtained objective value. As in the first test, 200 initial solutions are created from random permutations, which form the starting points for all heuristics. An interesting observation is that the hard time window constraints have different effects on the runtime of the procedures. Since a cheapest insertion heuristic is included within the chromosome crossover to obtain time window compliance, the evolutionary methods approximate more quickly. In contrast to the before mentioned methods, the LS procedures lead to increasing runtimes. Due to the fact that the solution space is significantly reduced by hard time windows, even Cplex is able to compute integer solutions for instance with 80 vertices. However, all these effects cannot be attributed solely to hard time windows, since the benchmark instances differ slightly in many points.

Table 4. Bredström and Rönnqvist [5] benchmarks and objective (15)

	F'	LB	cpu	F'^{GA}	cpu	F'^{hGA}	cpu	F'^{SDS}	cpu	F'^{MN}	cpu	F'^{AVN}	cpu
A'.0	**666.4**	681.2	<1	**666.4**	27	**666.4**	174	698.0	9	689.3	10	699.5	5
A'.1	**659.6**	659.6	<1	**659.6**	7	**659.6**	183	663.7	12	663.7	12	663.7	6
A'.2	**704.0**	704.0	<1	**704.0**	10	**704.0**	190	711.4	11	711.4	11	**704.0**	5
A'.3	**672.0**	672.0	<1	**672.0**	13	**672.0**	265	683.0	9	683.0	10	683.0	9
A'.4	**622.5**	622.5	<1	**622.5**	14	**622.5**	178	662.2	12	667.2	14	672.5	6
A'.5	**757.4**	757.4	<1	**757.4**	11	**757.4**	188	760.3	9	760.3	9	760.3	4
A'.6	**768.9**	768.9	<1	**768.9**	16	**768.9**	255	796.8	8	814.4	8	839.8	5
A'.7	**724.3**	724.3	<1	**724.3**	15	**724.3**	158	766.3	8	766.3	9	744.4	4
A'.8	**618.5**	618.5	<1	**618.5**	16	**618.5**	233	662.3	8	662.3	8	662.3	5
A'.9	**618.5**	618.5	<1	**618.5**	12	**618.5**	192	640.2	11	635.4	11	647.2	6
B'.0	**1141.3**	1026.2	36025	1171.8	128	1163.1	2230	1299.0	664	1308.8	907	1401.9	269
B'.1	1212.2	945.0	36000	1175.4	170	**1169.9**	3602	1330.1	878	1299.8	1100	1350.6	337
B'.2	**1367.6**	1227.2	36033	1390.4	201	1380.4	3608	1474.3	772	1497.9	1077	1523.7	315
B'.3	1258.9	1088.7	36000	1240.8	96	**1210.8**	3615	1344.5	790	1334.9	1084	1388.5	294
B'.4	**1158.5**	1119.3	36027	**1158.5**	199	1176.3	3606	1279.5	799	1268.6	900	1321.4	241
B'.5	**1276.8**	1157.5	36000	1287.7	164	1304.0	3603	1710.6	746	1423.2	1020	1467.7	281
B'.6	1285.3	1109.5	36000	1283.9	112	**1276.3**	3615	1411.8	768	1383.6	1084	1433.3	291
B'.7	**1144.4**	1047.1	36032	1158.7	129	**1144.4**	3602	1373.9	754	1308.4	1067	1314.7	303
B'.8	**1105.9**	1024.9	36021	1129.5	118	**1105.9**	3209	1684.9	720	1238.7	914	1300.7	281
B'.9	1112.6	804.2	36000	1059.9	170	**1043.1**	3601	1168.1	825	1141.3	1121	1215.9	344
C'.0	-	1207.4	36210	**1632.9**	427	2358.9	3663	1970.9	3681	2077.7	3921	2186.7	981
C'.1	-	1280.7	36026	**1616.9**	768	1795.2	3804	1924.0	3678	1978.7	3616	2137.4	824
C'.2	1727.5	1245.9	36107	**1742.9**	542	1903.6	3858	2040.1	3499	2032.7	3612	2284.8	955
C'.3	-	1086.6	36154	**1681.6**	687	1775.3	3677	1936.5	3607	1849.7	3617	1899.8	1084
C'.4	-	1145.5	36092	**1653.3**	705	1745.6	3822	1899.5	3600	1830.0	3605	1856.6	998
C'.5	-	1172.7	36003	**1657.3**	1079	1867.3	3782	1967.8	3488	1910.0	3617	2118.0	1072
C'.6	1959.6	1277.6	36055	**1830.6**	732	1942.7	3630	2042.3	3553	2058.7	3610	2030.1	950
C'.7	-	1169.4	36051	**1609.2**	617	1737.5	3811	1875.7	3376	1931.3	3613	1951.8	1037
C'.8	1848.8	1260.6	36003	**1780.2**	686	1953.1	3666	1934.4	3441	2006.5	3603	2225.4	1018
C'.9	-	1246.8	36045	**1635.7**	628	1829.7	3652	1981.8	3607	1890.8	3601	2038.1	1061

Moreover, the GA and hGA find the best objective values for all instances of (A') and (B'). However, the results of the Genetic-Algorithm F^{GA} are very

close to the entries of F^{hGA}. On the contrary the results show that Local Search procedures of Mankowska et al. [19] are less suitable for hard time windows. In instance type (C') the GA dominates all other procedures in terms of solution quality and runtime. For a problem size of 100 vertices, the GA proves to be the dominant method. The GA delivers both the highest solution quality and has the shortest computing time.

5.2 Sensitivity analysis

We investigate the influence of different weights of the objective terms on solution quality. In case of Mankowska et al. [19] benchmarks, the instances of type (B) are solved again for the sole minimization of the individual terms in equation (13). That means, minimizing the overall distance with $\gamma_1 = 1$ ($\gamma_2 = \gamma_3 = 0$), the accumulated tardiness with $\gamma_2 = 1$ ($\gamma_1 = \gamma_3 = 0$) or maximal observed tardiness with $\gamma_3 = 1$ ($\gamma_1 = \gamma_2 = 0$), respectively. The results are shown in table 5. Again F denotes the best obtained objective value of the associated solution procedure, LB is the lower bound determined by Cplex and cpu represents the computational time.

Table 5 shows that minimizing the distance term is the most difficult criterion to optimize for commercial solvers. As soon as time window violations are no longer penalized, the amount of nodes within the branch-and-bound tree significantly increases. The runtimes of the heuristic procedures remain almost unchanged in the first scenario compared to the main study. However, apart from the hGA, the solution quality of all heuristics decrease slightly when the distance is the only objective criterion. Once again, the hGA proves to be the best heuristic method for a VRPS with soft time window constraint for the latest possible start, providing enough available computing time.The standalone GA proves to be inferior to local search methods by Mankowska et al. [19]. Both, the overall tardiness and the maximum observed tardiness are considered to be relatively easy to minimize. Each instance is solved to an optimum by all procedures in a short amount of time.

As one can see from the results in Table 5, the aim to obtain time window compliance not matter what the price is, requires a much lower computational effort. This is because we reduced the size of the coordinate system of the Mankowska et al. [19] benchmarks until all patients or vertices, respectively, can be operated within one day. For a balance, we reduce the size of the time windows to 90 minutes. As it can be seen, the instances are now actually on the verge of admissibility even for hard time window constraints.

We proceed in the same way with the benchmark instances of type (A') according to Bredström and Rönnqvist [5]. Since the minimization of the distance was already part of the main study, the sensitivity analysis is limited to the minimization of the route duration $\lambda_2 = 1$ ($\lambda_1 = \lambda_3 = 0$) or vehicle fixed costs $\lambda_3 = 1$ ($\lambda_1 = \lambda_2 = 0$). The numerical results are presented in table 6.

Table 6 gives some interesting observations. As already mentioned, the optimization potential of the heuristics is limited by the simplified determination of the operation start time. But also on state-of-art solver, the consideration of tour

Table 5. Sensitivity analysis Mankowska et al. [19] and objective (13)

	Type	F	LB	cpu	F^{GA}	cpu	F^{hGA}	cpu	F^{SDS}	cpu	F^{MN}	cpu	F^{AVN}	cpu
Distance*	B.0	**877.7**	877.7	4135	960.9	62	**877.7**	519	945.9	31	943.2	36	991.2	26
	B.1	**616.8**	616.8	5773	655.0	40	**616.8**	288	708.0	34	655.0	34	691.5	27
	B.2	**951.1**	951.1	22770	1036.7	64	**951.1**	394	985.7	34	965.9	38	984.9	27
	B.3	**995.6**	995.6	9290	1029.5	199	**995.6**	441	1085.9	39	1044.7	41	1120.9	29
	B.4	**899.5**	899.5	460	949.4	114	**899.5**	336	970.7	25	916.3	28	972.4	20
	B.5	**967.0**	922.8	36023	1065.2	309	**967.0**	596	1043.5	39	1014.2	39	1029.6	29
	B.6	**745.3**	745.3	6075	868.4	91	**745.3**	216	815.1	39	789.9	42	799.0	29
	B.7	**882.9**	882.9	29600	1012.8	66	**882.9**	466	926.4	29	920.1	34	882.9	23
	B.8	**723.9**	723.9	5602	767.6	219	**723.9**	321	758.2	39	758.1	39	795.8	27
	B.9	**1005.0**	976.2	36001	1185.0	41	**1005.0**	271	1072.6	35	1068.9	36	1056.7	27
Tardiness**	B.0	**151.4**	151.4	132	**151.4**	21	**151.4**	65	**151.4**	28	**151.4**	32	**151.4**	26
	B.1	**0.0**	0.0	8	**0.0**	16	**0.0**	38	**0.0**	25	**0.0**	24	**0.0**	17
	B.2	**5.0**	5.0	56	**5.0**	17	**5.0**	41	**5.0**	29	**5.0**	29	**5.0**	19
	B.3	**0.0**	0.0	11	**0.0**	21	**0.0**	91	**0.0**	34	**0.0**	34	**0.0**	24
	B.4	**19.6**	19.6	27	**19.6**	17	**19.6**	42	**19.6**	24	**19.6**	25	**19.6**	19
	B.5	**0.0**	0.0	9	**0.0**	23	**0.0**	80	**0.0**	34	**0.0**	35	**0.0**	25
	B.6	**0.0**	0.0	10	**0.0**	21	**0.0**	100	**0.0**	34	**0.0**	39	**0.0**	35
	B.7	**0.0**	0.0	11	**0.0**	16	**0.0**	43	**0.0**	27	**0.0**	28	**0.0**	28
	B.8	**0.0**	0.0	25	**0.0**	19	**0.0**	16	**0.0**	7	**0.0**	35	**0.0**	35
	B.9	**0.0**	0.0	9	**0.0**	15	**0.0**	42	**0.0**	29	**0.0**	27	**0.0**	20
MaxTardiness***	B.0	**104.1**	104.1	35	**104.1**	20	**104.1**	67	**104.1**	7	**104.1**	33	**104.1**	23
	B.1	**0.0**	0.0	10	**0.0**	16	**0.0**	47	**0.0**	29	**0.0**	34	**0.0**	24
	B.2	**5.0**	5.0	23	**5.0**	17	**5.0**	54	**5.0**	8	**5.0**	41	**5.0**	26
	B.3	**0.0**	0.0	23	**0.0**	21	**0.0**	89	**0.0**	9	**0.0**	52	**0.0**	32
	B.4	**19.6**	19.6	28	**19.6**	17	**19.6**	42	**19.6**	35	**19.6**	37	**19.6**	26
	B.5	**0.0**	0.0	10	**0.0**	23	**0.0**	65	**0.0**	8	**0.0**	44	**0.0**	29
	B.6	**0.0**	0.0	10	**0.0**	19	**0.0**	94	**0.0**	7	**0.0**	53	**0.0**	30
	B.7	**0.0**	0.0	13	**0.0**	16	**0.0**	48	**0.0**	7	**0.0**	43	**0.0**	27
	B.8	**0.0**	0.0	15	**0.0**	15	**0.0**	66	**0.0**	6	**0.0**	37	**0.0**	22
	B.9	**0.0**	0.0	10	**0.0**	15	**0.0**	42	**0.0**	23	**0.0**	42	**0.0**	28

* $\gamma_1 = 1, \gamma_2 = \gamma_3 = 0$
** $\gamma_2 = 1, \gamma_1 = \gamma_3 = 0$
*** $\gamma_3 = 1, \gamma_1 = \gamma_2 = 0$

duration has a significant impact. However, finding good solutions is less difficult than testing for optimality. That means, the reduction of the relative gap is associated with a disproportional amount of time. In comparison to the main study, it becomes clear that minimizing the distance in a VRPS with hard time window constraints is a comparatively easy optimization problem. Since duration is the only objective term, all heuristics are far away from the optimum for the first time, however, the evolutionary algorithms obtain the higher solution quality. All mentioned heuristics are limited by the AH as construction method. For the minimization of route duration a more sophisticated construction method or repair algorithms are required.

Since at least two vehicles are required to proceed a synchronization vertex and the total amount of available vehicles in A' are 4 the problem to minimize the vehicle counter becomes trivial. However, none of the heuristic methods has an operator to reduce the number of vehicles. This is particularly evident in the LS. In 50 % of the cases the Local-Search is not able to find the optimum. This can be easily be justified because the LS procedure can only eliminate routes that

Table 6. Sensitivity analysis Bredström and Rönnqvist [5] and objective (15)

	Type	F'	LB	cpu	F'^{GA}	cpu	F'^{hGA}	cpu	F'^{SDS}	cpu	F'^{MN}	cpu	F'^{AVN}	cpu
Duration*	A'.0	**1239.6**	541.1	36001	1301.4	27	1301.4	405	1418.4	7	1399.8	8	1399.8	4
	A'.1	**1181.5**	544.4	36003	1262.6	28	1262.6	181	1343.8	6	1343.8	6	1369.5	3
	A'.2	**1378.9**	780.9	36005	1495.3	18	1495.3	342	1620.5	7	1620.5	8	1620.5	4
	A'.3	**1206.1**	346.3	36030	1366.1	27	1297.3	313	1366.1	5	1390.0	7	1390.0	4
	A'.4	**1227.8**	326.7	36021	1315.2	38	1315.2	430	1398.0	9	1398.0	9	1398.0	5
	A'.5	**1330.8**	766.9	36039	1382.2	13	1382.2	213	1451.4	6	1445.4	7	1426.5	4
	A'.6	**1345.7**	1209.5	36001	1445.3	36	1452.6	492	1547.6	5	1547.6	5	1644.3	4
	A'.7	**1262-9**	690.5	36007	1315.4	12	1308.8	139	1375.9	5	1375.9	6	1375.9	3
	A'.8	**1101.6**	548.2	36022	1224.5	14	1224.5	156	1385.9	5	1338.5	5	1421.5	3
	A'.9	**1105.0**	450.5	36035	1155.9	18	1155.3	348	1290.0	8	1290.0	8	1290.0	4
Vehicle Counter**	A'.0	**300.0**	300.0	5	**300.0**	15	**300.0**	90	**300.0**	2	**300.0**	2	**300.0**	2
	A'.1	**300.0**	300.0	24	**300.0**	10	**300.0**	43	400.0	2	400.0	2	400.0	2
	A'.2	**300.0**	300.0	6	**300.0**	11	**300.0**	44	**300.0**	2	**300.0**	2	**300.0**	2
	A'.3	**300.0**	300.0	21	**300.0**	13	**300.0**	90	400.0	2	400.0	2	400.0	2
	A'.4	**300.0**	300.0	4	**300.0**	11	**300.0**	44	**300.0**	2	**300.0**	2	**300.0**	2
	A'.5	**300.0**	300.0	34	**300.0**	14	**300.0**	86	400.0	2	400.0	2	400.0	2
	A'.6	**300.0**	300.0	3	**300.0**	13	**300.0**	44	**300.0**	2	**300.0**	2	**300.0**	2
	A'.7	**300.0**	300.0	7	**300.0**	16	**300.0**	91	400.0	2	400.0	2	400.0	2
	A'.8	**300.0**	300.0	23	**300.0**	6	**300.0**	45	400.0	2	400.0	2	400.0	2
	A'.9	**300.0**	300.0	89	**300.0**	13	**300.0**	92	**300.0**	2	**300.0**	2	**300.0**	2

* $\lambda_2 = 1, \lambda_1 = \lambda_3 = 0$
** $\lambda_3 = 1, \lambda_1 = \lambda_2 = 0$

contain only one vertex. If the initial solution does not contain such a route, the procedures stops after the first iteration, since there is no other objective term which can be optimized. Therefore, a minimization of the number of vehicles is pure coincidence. In principle, the same applies to the evolutionary methods, but due to the large population compared to the solution space in the scenario, they were able to determine the optimal solutions.

6 Conclusion

We successfully applied the first Genetic Algorithm to the VRP with temporal synchronization constraints using a permutation of the vertices as chromosome decoding. The presented method has been designed for both soft and hard time window constraints. In doing so, we introduced an efficient preprocessing for feasibility checks to obtain hard time window constraints. To evaluate the procedure, we compare it with the Local-Search procedures by Mankowska et al. [19]. As part of this, we also implement a hybrid metaheuristic hGA, which is a combination of GA and LS.

The results for soft time window constraints show that GA, especially the hybrid design, is a promising solution approach. For all instances, hGA determines the solution with the best objective value. These solutions always corresponded to the optimum, providing Cplex could determine this. For the other instances, the average savings compared to the LS is about 10 %. A disadvantage of the hGA is that the maximum problem size is limited to 100 vertices. The solution quality of the standalone GA can also keep up with the LS, but longer runtimes

are necessary. In addition, it turns out that the GA approximates very slowly for larger problem sizes (in case of soft time window constraints).

In case of hard time window constraints, the evolutionary metaheuristics GA and hGA dominates LS procedures. Even for instances with 50 vertices the deviation on average is not more than 5 % off the optimum. But even at 100 vertices, the solution quality of the evolutionary methods proved to be considerably higher than in the LS. Since cheapest insertion heuristic is used to obtain time window compliance if a violation occurs, the approximation of the GA is significantly increased. It is therefore possible that the GA can solve even larger instances within one hour, providing hard time window constraints.

Moreover, we uncovered a blind spot of heuristic procedure minimizing the duration for the VRPS. As far as we know, there are no operators specifically designed to minimize the duration of the tour. Even commercial solvers fail to minimize the route duration. Although the GA and hGA achieved the highest solution quality among the heuristics, but are far away from the optimum. Our goal is to try to tackle this problem in future research. In doing so, we intend to include a modification of the matheuristic by Bredström and Rönnqvist [6] in the computational study. Based on fixed routing variables, the solver should be able to reduce the tour duration, even if no optimality can be guaranteed.

According to the results presented, evolutionary methods appear to be quite promising. Similar procedures to the proposed GA using a traditional TSP chromosome encoding are possible. For decoding a AH introduced by Labadie et al. [15] or Li et al. [16] can be used. But also Particle-Swarm-Optimization, which to the best of our knowledge has not yet been applied to the VRPS, are conceivable for both solution representations.

References

1. Alberto, Amaya, André Langevin, and Martin Trépanier: *A heuristic method for the capacitated arc routing problem with refill points and multiple loads.* Journal of the Operational Research Society, 61(7):1095–1103, 2010.
2. Amaya, Alberto, André Langevin, and Martin Trépanier: *The Capacitated Arc Routing problem with refill points.* Operations Research Letters, 35(1):45 – 53, 2007, ISSN 0167-6377.
3. Baker, Barrie M. and M. A. Ayechew: *A genetic algorithm for the vehicle routing problem.* Computers & Operations Research, 30(5):787–800, April 2003, ISSN 0305-0548.
4. Bélanger, Nicolas, Guy Desaulniers, François Soumis, and Jacques Desrosiers: *Periodic Airline Fleet Assignment with Time Windows, Spacing Constraints, and Time Dependent Revenues.* European Journal of Operational Research, 175(3):1754 – 1766, 2006.
5. Bredström, David and Mikael Rönnqvist: *A branch and price algorithm for the combined vehicle routing and scheduling problem with synchronization constraints.* Discussion papers 2007/7, Department of Business and Management Science, Norwegian School of Economics, 2007.
6. Bredström, David and Mikael Rönnqvist: *Combined Vehicle Routing and Scheduling with Temporal Precedence and Synchronization Constraints.* European Journal of Operational Research, 191(1):19–31, 2008.

7. Cheng, Chi Bin and Keng Pin Wang: *Solving a Vehicle Routing Problem with Time Windows by a Decomposition Technique and a Genetic Algorithm.* Expert Systems with Application, 36(4):7758–7763, 2009.
8. Dantzig, Georg B. and J. H. Ramser: *The Truck Dispatching Problem.* Management Science, 6(1):80–91, October 1959.
9. Dohn, Anders, Esben Kolind, and Jens Clausen: *The Manpower Allocation Problem with Time Windows and Job-Teaming Constraints: A Branch-and-Price approach.* Computers & Operations Research, 36(4):1145–1157, 2009.
10. Drexl, Michael: *Synchronization in Vehicle Routing—A Survey of VRPs with Multiple Synchronization Constraints.* Transportation Science, 46(3):297–316, 2012.
11. Eveborn, Patrik, Patrik Flisberg, and Mikael Rönnqvist: *Laps Care—an operational system for staff planning of home care.* European Journal of Operational Research, 171(3):962 – 976, 2006.
12. Haddadene, S. R. Ait, Naciama Labadie, and Caroline Prodhon: *Grasp for the Vehicle Routing Problem with Time Windows, Synchronization and Precedence Constraints.* In *2014 IEEE 10th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pages 72–76, Oct 2014.
13. Ioachim, Irina, Jacques Desrosiers, Francois Soumis, and Nicolas Belanger: *Fleet Assignment and Routing with Schedule Synchronization Constraints.* European Journal of Operational Research, 119(1):75–90, 1999.
14. Kergosien, Yannick, Christophe Lenté, and Jean Charles Billaut: *An Extended Multiple Traveling Salesman Problem.* In *Multidisciplinary International Conference on Scheduling : Theory and Applications (MISTA 2009) 10-12*, 2009.
15. Labadie, Nacima, Christian Prins, and Yanyan Yang: *Iterated Local Search for a Vehicle Routing Problem with Synchronization constraints.* In *ICORES 2014 - Proceedings of the 3rd International Conference on Operations Research and Enterprise Systems*, pages 257–263, 2014.
16. Li, Yanzhi, Andrew Lim, and Brian Rodrigues: *Manpower allocation with time windows and job-teaming constraints.* Naval Research Logistics, 52(4):302–311, June 2005, ISSN 0894-069X.
17. Lim, Andrew, Brian Rodrigues, and L Song: *Manpower allocation with time windows.* Journal of the Operational Research Society, 55(11):1178–1186, 2004.
18. Liu, Ran, Xiaolan Xie, Vincent Augusto, and Carlos Rodriguez: *Heuristic algorithms for a vehicle routing problem with simultaneous delivery and pickup and time windows in home health care.* European Journal of Operational Research, 230(3):475–486, 2013.
19. Mankowska, Dorota S., Frank Meisel, and Christian Bierwirth: *The Home Health Care Routing and Scheduling Problem with Interdependent Services.* Health Care Management Science, 17(1):15–30, 2014.
20. Masson, Renaud, Fabien Lehuédé, and Olivier Péton: *Efficient Feasibility Testing for Request Insertion in the Pickup and Delivery Problem with Transfers.* Operations Research Letters, 41(3):211–215, 2013.
21. Potvin, Jean Yves and Samy Bengio: *The vehicle routing problem with time windows part ii: Genetic search.* INFORMS Journal on Computing, 8(2):165–172, 1996.
22. Prins, Christian: *A simple and effective Evolutionary Algorithm for the Vehicle Routing Problem.* Computers & Operations Research, 31(12):1985–2002, 2004.
23. Rasmussen, Matias Sevel, Tor Justesen, Anders Dohn, and Jesper Larsen: *The home care crew scheduling problem: Preference-based visit clustering and temporal dependencies.* European Journal of Operational Research, 219(3):598 – 610, 2012.

24. Rousseau, Louis Martin, Michel Gendreau, Gilles Pesant, Cp Succ Centre-ville, and Hc J: *The Synchronized Dynamic Vehicle Dispatching Problem.* INFOR Information Systems and Operational Research, 51, December 2002.
25. Salazar-Aguilar, Angélica, André Langevin, and Gilbert Laporte: *Synchronized Arc Routing for snow plowing operations.* Computers & Operations Research, 39:1432–1440, July 2012.
26. Savelsbergh, Martin W. P.: *Local Search in Routing Problems with Time Windows.* Annals of Operations Research, 4(1):285–305, 1985.
27. Scheffler, Martin, Christina Hermann, and Mathias Kasper: *Splitting procedure of genetic algorithm for column generation to solve a vehicle routing problem.* In Fink, Andreas, Armin Fügenschuh, and Martin Josef Geiger (editors): *Operations Research Proceedings 2016: Selected Papers of the Annual International Conference of the German Operations Research Society (GOR), Helmut Schmidt University Hamburg, Germany, August 30 - September 2, 2016*, pages 321–328. Springer International Publishing, 2017.
28. Schönberger, Jörn: *Operational Freight Carrier Planning: Basic Concepts, Optimization Models and Advanced Memetic Algorithms.* GOR-Publications. Springer Berlin Heidelberg, 2005, ISBN 9783540253181.
29. Schönberger, Jörn: *Implicit Time Windows and Multi-Commodity Mixed-Fleet Vehicle Routing.* Diskussionsbeiträge aus dem Institut für Wirtschaft und Verkehr, (1/2017), 2017.
30. Solomon, Marius M.: *Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints.* Operations Research, 35(2):254–265, April 1987, ISSN 0030-364X.
31. Thangiah, S. R., K. E. Nygard, and P. L. Juell: *Gideon: a genetic algorithm system for vehicle routing with time windows.* In *[1991] Proceedings. The Seventh IEEE Conference on Artificial Intelligence Application*, volume i, pages 322–328, Feb 1991.
32. Thangiah, S.R.: *Vehicle routing with time windows using genetic algorithms.* Technical report, Computer Science Department, Slippery Rock University, 1993.
33. Vidal, Thibaut, Teodor Gabriel Crainic, Michel Gendreau, and Christian Prins: *A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows.* Computers & Operations Research, 40(1):475 – 489, 2013, ISSN 0305-0548.

SEIT 2000 SIND FOLGENDE DISKUSSIONSBEITRÄGE ERSCHIENEN:

1/2000 Röhl, Klaus-Heiner: Die Eignung der sächsischen Agglomerationsräume als Innovations- und Wachstumspole für die wirtschaftliche Entwicklung des Landes

2/2000 Röhl, Klaus-Heiner: Der Aufbau der ostdeutschen Infrastruktur und sein Beitrag zur wirtschaftlichen Entwicklung in Sachsen

3/2000 Kummer, Sebastian; Mating, Anette; Käsbauer, Markus; Einbock, Marcus: Franchising bei Verkehrsbetrieben

4/2000 Westphal, Jan R.: Komplexitätsmanagement in der Produktionslogistik

5/2000 Röhl, Klaus-Heiner: Saxony's Capital Dresden – on the Way to become Eastern Germany's first "Innovative Milieu"?

6/2000 Schramm, Hans-Joachim: Electronic Commerce im Lebensmitteleinzelhandel - Auswertung einer Konsumentenbefragung im Großraum Dresden

1/2001 Schramm, Hans-Joachim; Veith, Elisabeth: Schwerlasttransport auf deutschen Straßen, Ergebnisse einer Befragung deutscher Schwerlasttransportunternehmen

2/2001 Schramm, Hans-Joachim; Eberl, Katharina: Privatisierung und Going Public von staatlichen Eisenbahnunternehmen - Versuch eines adaptiven Vergleichs zwischen Japan und Deutschland

1/2002 Kummer, Sebastian; Schmidt, Silvia: Methodik der Generierung und Anwendung wertorientierter Performance-Kennzahlen zur Beurteilung der Entwicklung des Unternehmenswertes von Flughafenunternehmen

2/2002 Wieland, Bernhard: Economic and Ecological Sustainability - The Identity of Opposites?

1/2003 Freyer, Walter; Groß, Sven: Tourismus und Verkehr - Die Wechselwirkungen von mobilitätsrelevanten Ansprüchen von touristisch Reisenden und Angeboten (touristischer) Transportunternehmen

2/2003 Stopka, Ulrike; Urban, Thomas: Implikationen neuer Vertriebs- und Distributionsformen auf das Customer Relationship Management und die Gestaltung von virtuellen Marktplätzen im BtoC-Bereich

1/2004 Hoppe, Mirko; Schramm, Hans-Joachim: Use of Interorganisational Systems - An Empirical Analysis

2/2004 Wieland, Bernhard; Seidel, Tina; Matthes, Andreas; Schlag, Bernhard: Transport Policy, Acceptance and the Media

1/2005 Brunow, Stephan; Hirte, Georg: Age Structure and Regional Income Growth

2/2005 Stopka, Ulrike; Urban, Thomas: Erklärungsmodell zur Beurteilung der betriebswirtschaftlichen Vorteilhaftigkeit des Kundenbeziehungsmanagements sowie Untersuchung zur Usability von Online-Angeboten im elektronischen Retailbanking

3/2005 Urban, Thomas: Medienökonomie

4/2005 Urban, Thomas: eMerging-Media: Entwicklung der zukünftigen Kommunikations- und Medienlandschaft

1/2006 Wieland, Bernhard: Special Interest Groups and 4th Best Transport Pricing

2/2006 Ammoser, Hendrik; Hoppe, Mirko: Glossar Verkehrswesen und Verkehrswissenschaften

1/2007 Wieland, Bernhard: Laudatio zur Verleihung der Ehrendoktorwürde an Herrn Prof. Dr. rer. pol. habil. Gerd Aberle

2/2007 Müller, Sven; Kless, Sascha: Veränderung der leistungsabhängigen Schwerverkehrsabgabe in Abhängigkeit der Streckenbelastung

1/2008 Vetter, Thomas; Haase, Knut: Alternative Bedienformen im ÖPNV – Akzeptanzstudie im Landkreis Saalkreis

2/2008 Haase, Knut; Hoppe, Mirko: Standortplanung unter Wettbewerb – Teil 1: Grundlagen

3/2008 Haase, Knut; Hoppe, Mirko: Standortplanung unter Wettbewerb – Teil 2: Integration diskreter Wahlentscheidungen

1/2009 Günthel, Dennis; Sturm, Lars; Gärtner, Christoph: Anwendung der Choice-Based-Conjoint-Analyse zur Prognose von Kaufentscheidungen im ÖPNV

2/2009 Müller, Sven: A Spatial Choice Model Based on Random Utility

1/2010 Lämmer, Stefan: Stabilitätsprobleme voll-verkehrsabhängiger Lichtsignal-steuerungen

2/2010 Evangelinos, Christos; Stangl, Jacqueline: Das Preissetzungsverhalten von Fluggesellschaften auf Kurzstrecken mit Duopolcharakter

3/2010 Evangelinos, Christos; Matthes, Andreas; Lösch, Stefanie; Hofmann, Maria: Parking Cash Out – Ein innovativer Ansatz zur betrieblichen Effizienzsteigerung und Verkehrslenkung

1/2011 Evangelinos, Christos; Püschel, Ronny; Goldhahn Susan: Inverting the Regulatory Rules? Optimizing Airport Regulation to Account for Commercial Revenues

2/2011 Evangelinos, Christos; Obermeyer, Andy; Püschel, Ronny: Preisdispersion und Wettbewerb im Luftverkehr – Ein theoretischer und empirischer Überblick

1/2012 Geller, Kathleen; Evangelinos, Christos; Hesse, Claudia; Püschel, Ronny; Obermeyer, Andy: Potentiale und Wirkungen des EuroCombi in Deutschland

2/2012 Deweß, Sigrun; Klier, Michael: Verfahren zur Beschränkung von Schwerpunktmodulplätzen am Institut für Wirtschaft und Verkehr

1/2013 Evangelinos, Christos: Infrastrukturpreise - Eine normativ-theoretische Analyse

2/2013 Evangelinos, Christos: Interessengruppen und Preissetzung im Verkehr

1/2014 Hermann, Christina: Die kombinierte Touren- und Personaleinsatzplanung von Pflegediensten – Teil 1: Literatur und Modell

2/2014 Hirte, Georg; Stephan, Andreas: Regionale Beschäftigungswirkungen von öffentlichen Investitionen in Straßen- und Schieneninfrastruktur

1/2015 Schönberger, Jörn: Vehicle Routing with Source Selection - Integrating Sourcing in Fleet Deployment

2/2015 Schönberger, Jörn: Scheduling of Sport League Systems with Inter-League Constraints

3/2015 Schönberger, Jörn: Hybrid Search and the Dial-A-Ride Problem with Transfer Scheduling Constraints

1/2016 Hermann, Christina: Die kombinierte Touren- und Personaleinsatzplanung von Pflegediensten – Teil 2: Ergebnisse

2/2016 Evangelinos, Christos; Wittkowski, Antje; Püschel, Ronny: A Note on "Price Cap Regulation of Congested Airports"

1/2017 Schönberger, Jörn: Implicit Time Windows and Multi-Commodity Mixed-Fleet Vehicle Routing

2/2017 Hocke, Stephan; Gajewski, Christina; Kasper, Mathias: A Genetic Algorithm for Vehicle Routing Problems with Temporal Synchronization Constraints