Central European Journal of Operations Research (2021) 29:247–289
https://doi.org/10.1007/s10100-019-00625-0

A metaheuristic algorithm and structured analysis for the Line-haul Feeder Vehicle Routing Problem with Time Windows

Christian Brandstätter[1]

Published online: 15 May 2019

Abstract

Synchronisation in vehicle routing is a rather new field of research and naturally new problems arise. One of these problems is the Line-haul Feeder Vehicle Routing Problem (LFVRP). It uses a fleet of small and large vehicles to serve two types of customers. The first type provides additional parking space and can be visited by both vehicle classes. The second type can only be visited by the small vehicle class as these customers provide only limited parking space. The main characteristic of the small vehicle class is the limited capacity. To overcome this particular disadvantage, the small vehicles can use the large vehicles as virtual depots. In other words, a small and large vehicle can meet at a parking lot or at a customer with enough space (type-1 customer) and perform a transfer of goods. For a successful reloading operation, both vehicles must be present at the same place at the same time. Thus, both vehicle tours must be synchronized. After using the large vehicle as virtual depot, the small vehicle can proceed immediately afterwards because it does not need to go back to the physical depot. Consequently, less time and distance is required which results in a reduction of the overall costs. The advantage of the LFVRP over classical variants of the Vehicle Routing Problem has been shown in previous papers. Yet, customer time windows have been neglected so far and as time windows play an important role in vehicle routing research, they need to be addressed properly. Therefore, we aim to close this gap by introducing the Line-haul Feeder Vehicle Routing Problem with Time Windows (LFVRPTW). We discuss the complexity of customer time windows for the LFVRPTW and adopt the previously introduced algorithm for the LFVRP. Furthermore, we provide a thorough computational analysis on the impact of different time window characteristics and show the advantage of the LFVRPTW over other variants of the Vehicle Routing Problem with Time Windows.

✉ Christian Brandstätter
11brandc@gmail.com

[1] Department of Production and Operations Management, University of Graz, Graz, Austria

Keywords Vehicle routing problem · Synchronisation · Ant colony optimisation · Metaheuristics · Local search · Matheuristics · Transshipment · Time windows

1 Introduction

The digital age provides many new possibilities in city logistics and has already changed our way of life.[1] Nowadays, customers can look-up a large variety of goods on their mobile device and they can place an order whenever they like. Due to the vast amount of data, the customer is provided with all the necessary information of a potential purchase; e.g. pictures, number of available items, storage location, overall costs (purchase and delivery) and estimated delivery time. However, these new possibilities also increase the expectations of the customers. For instance: when ordering food, the customer expects the food to be delivered within minutes. Consequently, these customers are interested in the delivery status, current location and expected delivery time of their goods. In addition to the increased customer expectations, other problems arise as well. According to the United Nations (2014), the population of the major European cities will rise to their limits—population will increase up to 20% and more from 2005 to 2020. This will result in multiple issues like higher demands, increased traffic (especially during rush hour), rising land prices, more congested areas with limited space and many more. Companies specialised in city logistics have to master these challenges to keep their competitive edge. Consequently, new problems arise and one of these problems is the Line-haul Feeder Vehicle Routing Problem (LFVRP).

The LFVRP uses one physical depot (PD) where all vehicles are dispatched. To serve the customers, two classes of vehicles are used—the small (SV) and large (LV) vehicle class. The small vehicle class can be represented by e.g. a car, motorbike or cargo-bike. Yet, cargo-bikes are the preferred choice due to economic and environmental reasons. The costs for a cargo-bike are much less than e.g. a car, they can easily move around in congested areas and they to not have any emissions due to the absence of a combustion engine. However, vehicles with alternative power engines such as electric cars or motorbikes should be considered as well. To sum up, the characteristics of the small vehicle class are limited capacity, limited range and lesser costs. Compared to the small vehicle class, the large vehicle class has hardly any capacity limit (e.g. a truck with a trailer). Furthermore, they can drive longer distances but also have higher overall costs. As for the customers, they are classified according to their availability of parking space. Some customers (type-1) provide enough space and can be visited by both vehicle classes. Yet, the majority of customers (type-2) live in congested areas and are difficult to reach. Hence, only the small vehicle class can be used. Although the manoeuvre possibilities (especially in narrow streets and during rush hour) of the small vehicles are their biggest advantage, the main disadvantage of limited capacity remains. In other words, the small vehicle may only be able to visit a few customers before it must return to the PD. This necessary journey to the PD results

[1] City Logistics: The future of last mile delivery: 10 most important trends. http://www.citylogistics.info/food-for-thoughts/the-future-of-last-mile-delivery-10-most-important-trends/, Accessed: 26.12.2017.

in additional travelling costs and wasted time. To solve this issue, the small vehicle can use the large vehicle as a virtual depot (VD). This can happen at a parking place or at a customer with enough parking space (customer of type-1). However, the small and large vehicle need to synchronise their tours to be able to perform a transshipment. In other words, the small and large vehicle need to be at the same place at the same time. Thus, the small vehicle saves time and distance by avoiding travelling back to the PD. Furthermore, the overall costs will further decrease as the required number of small vehicles can be significantly reduced.

Although we extensively investigated the LFVRP in our previous papers, we neglected time windows so far. Our analysis showed that the problem is already very complex even without time windows. Early investigations supported the assumption that synchronisation between vehicles is hardly possible if time windows are in place. However, time windows play an important role in vehicle routing research and therefore need to be addressed accordingly. Therefore, we aim to close this gap by introducing the Line-haul Feeder Vehicle Routing Problem with Time Windows (LFVRPTW).

The contribution of this paper is twofold. First, we introduce time windows and adopt the well performing LFVRP-algorithm to handle time windows for customer service. Second, we analyse different time window categories and provide evidence that the LFVRPTW is still of advantage even with time windows in place. The remainder of this paper is structured as follows. In Sect. 2 we will review the existing literature on the LFVRP(TW) and closely related variants. A structural analysis on the impact of time windows will be given in Sect. 3. Section 4 will provide an overview of the used heuristic and describe each step in more detail. After that, a computational analysis is conducted in Sect. 5 where we will investigate different time window categories over large number of instances and compare our results to the best-known LFVRPTW results. The paper will be concluded with our final remarks in Sect. 6.

2 Literature review

The LFVRPTW is a rather new problem but at its core it is still a Vehicle Routing Problem (VRP). Furthermore, the solution approach features some well investigated and successfully applied heuristics. Therefore, we decided to provide a short yet broad overview of the relevant literature. We start with a generic overview of the VRP and the importance of time windows. Afterwards, we discuss relevant solution heuristics before we focus on relevant literature on the LFVRP(TW). We conclude the literature review by presenting similar problems with a heterogeneous fleet, usage of multiple tours and a specific customer vehicle relationship.

The VRP has been extensively studied over the last half century and is still a prominent topic within operations research. Hence, a vast number of literature on the different variants exist and for a solid overview the interested reader is advised to see Cordeau et al. (2002), Laporte (2009) or Laporte et al. (2014). Time windows play an important role in vehicle routing research and consequently most of the VRP variants have been analysed with time windows. To indicate an analysis with time windows, the suffix TW is usually added to the problem abbreviation (e.g. VRPTW).

Time windows are associated with the customers for most VRPTW variants. In other words, the vehicle must arrive in a certain time frame to start the service for the customer. However, other time windows like opening hours of shops, delivery time windows (e.g. before midday or night delivery) or driving bans for trucks due to particulate matter pollution may also be considered. Overall, time windows reduce the degree of freedom and increase the complexity of the VRP itself. If time windows (especially for delivery) are too tight, the effectiveness of the routing will be reduced significantly (see Russo and Comi 2010). An interesting research on the impact of time windows on city logistics applications can be found in Deflorio et al. (2012).

An early yet often used heuristic for handling time windows is the Push-Forward-Insertion-Heuristic (PFIH). It was originally introduced by Solomon (1987) and is an efficient heuristic for inserting a customer into an existing route while preserving time window constraints. Further reading on the PFIH can be found in Thangiah et al. (1993). Furthermore, a thorough survey on the VRPTW for route construction, local search algorithms and metaheuristics is provided in Bräysy and Gendreau (2005a) and Bräysy and Gendreau (2005b).

In early research papers on the VRP and its variants, problem specific heuristics were developed from scratch. These heuristics provided reasonably good results for the problem at hand but often failed if applied to other VRP variants. To overcome this disadvantage, generic solution heuristics—called metaheuristics—were developed and they can be applied to a large number of optimization problems. The most prominent representatives of metaheuristics are Ant Colony Optimization (ACO—e.g. Reimann et al. 2004; Dorigo and Blum 2005), Genetic Algorithms (GA—see Baker and Ayechew 2003; Prins 2004), Greedy Randomized Adaptive Search Procedure (GRASP—compare Feo and Resende 1995; Resende and Ribeiro 2003), Simulated Annealing (SA—e.g. Kirkpatrick et al. 1983; van Breedam 1995), Adaptive Large Neighbourhood Search (ALNS—see Ropke and Pisinger 2006a, b; Pisinger and Ropke 2007, 2010) and Tabu Search (TS—compare Glover 1989, 1990). For a solid overview on metaheuristics the interested reader is advised to see Gendreau et al. (2008) or Laporte (2009).

Although the CPU performance of modern computers is improved rapidly, the optimal VRP solution for large scaled problem sizes can hardly be found within a reasonable amount of time. Remark: the VRP and its variants are np-hard. However, for smaller scaled problem sizes the optimal solution can be found by the use of a mathematical problem solver. Thus, hybrid algorithms are developed where large sized problems are partially solved to optimality. This research field is known as matheuristics in the literature and the interested reader is advised to see Boschetti et al. (2009), Doerner and Schmid (2010), Subramanian et al. (2012) and Archetti and Speranza (2014).

The LFVRP originated from Asia/Taiwan and occurred during the delivery of lunch boxes. It was first introduced as the LFVRP with Virtual Depots (LFVRPVD) by Chen et al. (2011b). They proposed two simple heuristics and examined them on 17 test instances. In their second paper Chen et al. (2011a), they added time windows (LFVRPVDTW) and introduced a new (two-stage) heuristic which also included a tabu search procedure. Chen and Wang (2012) relaxed some of the limitations made by previous authors and also introduced a new construction heuristic—named the general insertion method—to improve the heuristic even further. For their final paper (see Chen 2015), several issues for the LFVRPTW have been analysed. These issues were different solution algorithms, adjustments to customer demands, number of type-1 customers (VDs) and relaxed time windows.

In Brandstätter and Reimann (2018b) we provided the first mathematical problem formulation for the LFVRP and introduced two new promising heuristics named the *Linkage-* and *Split-Approach*. Both heuristics provided significantly better results than the results presented by Chen et al. (2011b). Furthermore, we performed a sensitivity analysis and showed that the LFVRP benefits from a more decentralised depot, larger capacity for the small vehicle class, shorter service/transshipment times and more possibilities for reloading. We further improved the two proposed heuristics in Brandstätter and Reimann (2018a) by combining the advantages of metaheuristics, matheuristics, generating multiple solutions and local search. We were able to realise an improvement of around 9% compared to previous results.

The LFVRP(TW) has some similarities to already existing problems. One similarity is the heterogeneous fleet of vehicles. Most of the VRP(TW) problems assume a homogeneous fleet although a heterogeneous fleet is most often the case in real life applications. A heterogeneous fleet of vehicles has at least two or more vehicle classes which are classified according to certain characteristics (e.g. average speed, costs, capacity). In the literature this problem is often referred to as the Heterogeneous Fleet VRP (HFVRP (TW); compare e.g. Baldacci et al. 2008; Subramanian et al. 2012; Penna et al. 2013; Kritikos and Ioannou 2013; Koç et al. 2016); or in earlier publications as Fleet Size Mix (FSM: see Golden et al. 1984; Liu and Shen 1999).

Another characteristic of the LFVRP(TW) is the possibility of multiple tours per vehicle. In the classical formulation of the VRP(TW), a vehicle is used for a single tour only. However, if the capacity of a vehicle is depleted, it can drive back to the depot for reloading and continue afterwards with another tour. In the literature this problem is often referred as Multi Trip VRP, VRP with multiple routes, VRP with multiple use of vehicles or VRP with Multiple Trips (MTVRP) and a solid overview is provided by Cattaruzza et al. (2014), Cheikh et al. (2015), François et al. (2016) or Cattaruzza et al. (2016).

A rather closely related problem is the Site Dependent Vehicle Routing Problem (SDVRP). In this problem some customers can be only visited by a certain vehicle class. Hence, this problem also uses a heterogeneous fleet of vehicles and has a predefined customer-vehicle relationship. Further reading on the SDVRP can be found in Pisinger and Ropke (2007), Chao et al. (2016) and Cordeau et al. (2016). The main difference between the SDVRP and LFVRP(TW) is the synchronisation possibility. The synchronisation between vehicles is a rather young field of research where the vehicle tours are no longer independent. For example, two vehicles meet and the workers perform a certain task together—e.g. a transshipment of goods or a service/maintenance task where multiple service workers are needed. In recent years multiple problems with synchronisation constraints have been introduced (see Bredström and Rönnqvist 2008; Grangier et al. 2016; Anderluh et al. 2017). A lately conducted survey on vehicle routing problems with synchronisation constraints can be found in Drexl (2012).

3 Structural problem analysis

3.1 Review of previous work without time windows

In Brandstätter and Reimann (2018b) we presented two simple heuristic approaches named the *Linkage*- and *Split-Approach*. The basic concept of the *Linkage-Approach* is to generate feasible small vehicle tours for all type-2 customers and try to link them through a VD (type-1 customer) or the PD. If two tours can be linked through a VD, that VD must also be visited by a large vehicle at the same time to perform a transshipment of goods. Hence, the two tours need to be synchronised. The *Split-Approach*, on the other hand, generates a giant tour for all type-2 customers and tries to create feasible sub-tours by splitting the giant tour and inserting VDs or the PD into these sub-tours. If VDs are required to create feasible tours, these VDs must also be part of a large vehicle tour - like in the *Linkage-Approach*. The overall objective is to minimise the total cost consisting of fixed vehicle cost as well as variable fuel and wage costs for drivers.[2] Both approaches performed well but still had room for improvement. Therefore, we improved both approaches by combining the benefits of metaheuristics, matheuristics, generating multiple solutions and local search. A detailed overview can be found in Brandstätter and Reimann (2018a). Both approaches could be significantly improved, yet the *Linkage-Approach* outperformed the *Split-Approach* in every aspect. As a result, we decided to only proceed further with the *Linkage-Approach*. A detailed description of the different steps for the *Linkage-Approach* together with a small pseudo code will be provided in Sect. 4. For the following analysis, we will focus on the basic concept of the *Linkage-Approach* which can be described in four steps.

Step 1: Create small vehicle tours for all type-2 customers.
Step 2: Try to link the small vehicle tours through a VD or the PD to reduce the number of vehicles.
Step 3: Create large vehicle tours with used VDs in chronological order.
Step 4: Insert all remaining type-1 customers into the existing tours.

3.2 Problem analysis without time windows

Before we start with the problem analysis with time windows, we briefly review the characteristics of the *Linkage-Approach* without time windows. For a better understanding, we provide a small illustrative example in Fig. 1. The upper part (a) shows the result of the first step: two small vehicle tours with two type-2 customers (circle shape) each and two not yet routed customers of type-1 (diamond shape) which start and end at the depot (square shape with label PD).[3] Naturally, both tours start at the beginning of the time horizon as no further time restrictions concerning service exist.

As mentioned before, the variable part of the objective function includes the drivers wage. Thus, from a cost perspective it does not matter when the tours start as only the tour durations (t^d) are taken into account. The tour duration is calculated as tour

[2] For a more detailed objective function refer to Sect. 5.1.

[3] We will follow the figure notation of the depot and customers throughout the paper.

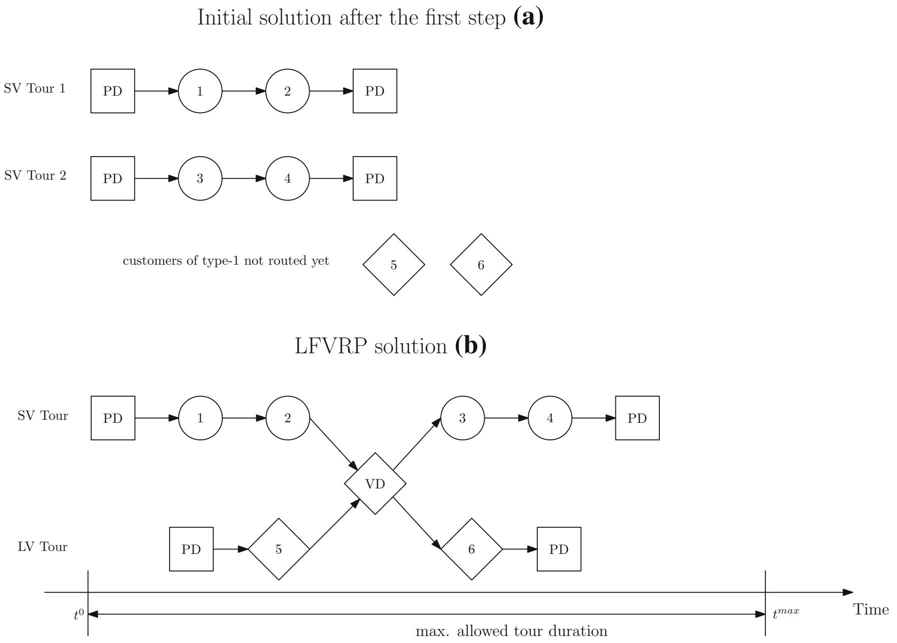

Initial solution after the first step (a)
SV Tour 1
PD
1
2
PD
SV Tour 2
PD
3
4
PD
customers of type-1 not routed yet
5
6
LFVRP solution (b)
SV Tour
PD
1
2
3
4
PD
VD
LV Tour
PD
5
6
PD
t^0
t^{max}
Time
max. allowed tour duration

Fig. 1 LFVRP example

finish time (t^{ft}) minus tour start time (t^{st}): $t^d = t^{ft} - t^{st}$. Therefore, if for example the starting time of SV Tour 2 is postponed, the cost value will remain the same as the finish time will be postponed as well. If we now proceed with the second step of the *Linkage-Approach*, these two tours will be linked through a VD. This is possible because the total tour duration of the resulting tour is still within the max. allowed tour duration. Hence, we have to replace the edges $e_{2,PD}$ with $e_{2,VD}$ of SV Tour 1 and $e_{PD,3}$ with $e_{VD,3}$ of SV Tour 2. In other words, customers 3 and 4 will also be served by the same small vehicle but at a later point in time. In our example the two remaining customers of type-1 will be served by the large vehicle (see Fig. 1b).

3.3 Problem analysis with time windows

Time windows represent an important aspect in vehicle routing research but also increase the complexity of the problem. The PFIH (introduced by Solomon 1987 and further described in Thangiah et al. 1993) is an effective heuristic for inserting a customer into an existing tour. However, in the context of the LFVRPTW we focus on moving a whole tour forwards or backwards in time. Therefore, we used the overall idea of the PFIH and adjusted it to our needs. For a better understanding we provide another small illustrative example in Fig. 2. The horizontal axis shows the time horizon whereas the customer numbers are displayed on the vertical axis. For simplification we show only the time window frames of the respective customers (grey areas) and the elapsed time concerning driving and service time. Figure X shows a single small vehicle tour

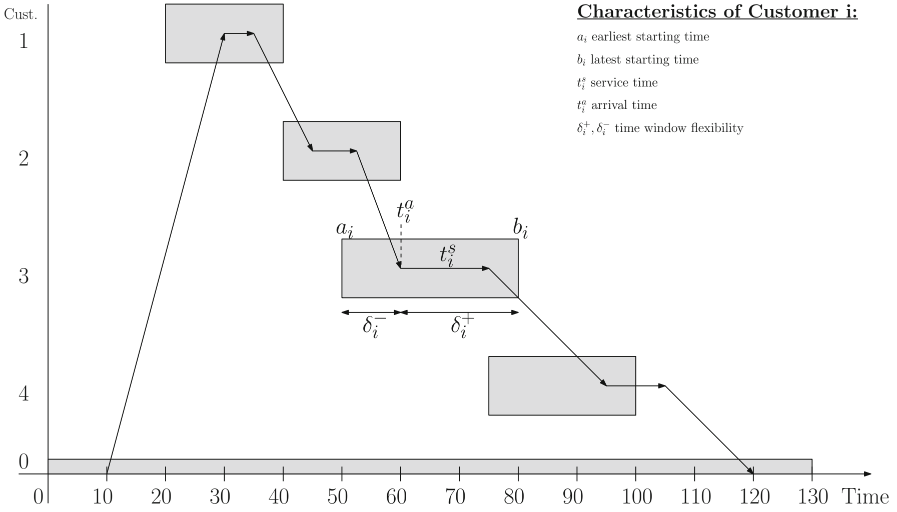


Fig. 2 Time windows analysis

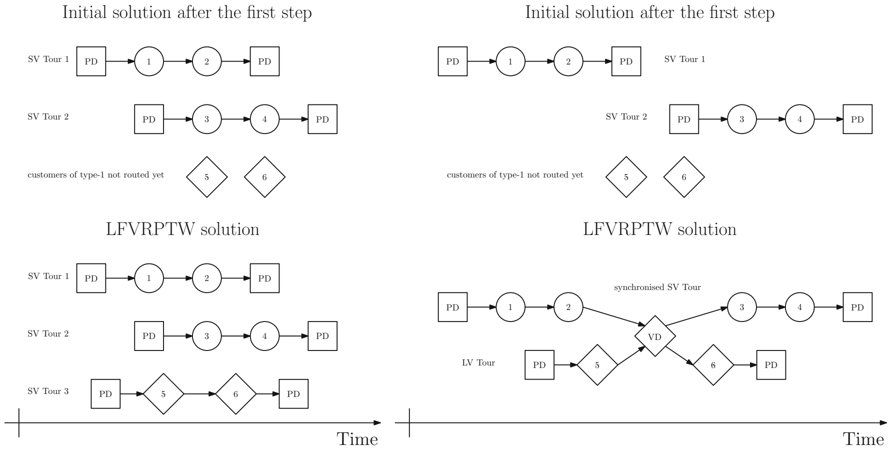
with four customers: 0 → 1 → 2 → 3 → 4 → 0. The time windows are denoted by (a_i) for the earliest starting time and (b_i) for the latest starting time of customer i. In other words, if the vehicle arrives (t_i^a) within this time frame ($a_i \leq t_i^a \leq b_i$), the service of this customer can start immediately. Yet, if the vehicle arrives before the time window starts ($t_i^a < a_i$), the vehicle has to wait until the time window opens. In addition, if the vehicle arrives after the time window closes ($b_i < t_i^a$), the customer cannot be served by this vehicle.

Figure 2 shows the small vehicle tour in the context of space and time. The vehicle tour starts at 10 and arrives within the time window of each customer and ends at 120 units. As all arrival times are within the time windows, no waiting times occur. Even the time window of the depot closes 10 units after the vehicle arrives. Hence, that tour can still be moved forwards in time without changing the total costs and also remains feasible. To find out how much the tour can be moved in time, we introduce a new variable called time window flexibility ($\delta_i^{+/-}$). The time window flexibility (δ_i^+) is calculated as the latest starting time minus the arrival time [see Eq. (1)] and (δ_i^-) as the arrival time minus earliest starting time [see Eq. (2)], respectively. (δ_i^+) denotes a move forwards and (δ_i^-) backwards in time and S the set of nodes/customers visited by the vehicle.

$$\delta_i^+ = b_i - t_i^a \tag{1}$$

$$\delta_i^- = \begin{cases} t_i^a - a_i & \text{if } \; t_i^a > a_i \\ 0 & \text{else} \end{cases} \tag{2}$$

$$\begin{gathered} \Delta_{max}^{+/-} = min(\delta_0^{+/-}, \ldots, \delta_i^{+/-}) \\ \forall i \in S \text{ and } S = \{0, \ldots, n\} \end{gathered} \tag{3}$$



(a) Synchronisation with TW is not possible **(b)** Synchronisation with TW is possible

Fig. 3 LFVRPTW example

As we can see in Fig. 2, some customers have larger ($\delta_i^{+/-}$) values than others. Thus, we have to find the minimum ($\delta_i^{+/-}$) to know the maximum amount of time ($\varDelta_{max}^{+/-}$) our tour can be moved forwards or backwards in time [see Eq. (3)]. In our example, the smallest (δ_i^{+}) is found at customer 4 with ($\varDelta_{max}^{+} = 5$ units) and (δ_i^{-}) at customer 2 with ($\varDelta_{max}^{-} = 5$ units).

If we use our first example again, but with time windows, we get the following situation: In Fig. 3 two possible scenarios are presented. Two small vehicle tours are positioned according to their chronological sequence and two customers of type-1 are not yet assigned. Customers in Fig. 3a have very close time windows (see customer 2 and 3), whereas the time windows in Fig. 3b are further apart. Furthermore, we assume that ($\varDelta_{max}^{+/-} = 0$), which means no further service postponement is possible. Consequently, if we further proceed with the *Linkage-Approach* we have two different outcomes. Due to the time window restriction, the LFVRPTW algorithm is able to reduce the number of small vehicles by linking the two small vehicle tours in example (b). In example (a) it is not possible to link the small vehicle tours together as customers 2 and 3 are required to be serviced at the same time. Overall, the solution in example (a) requires three small vehicles compared to example (b) with only one small and one large vehicle.

Note that in the LFVRPTW context, time windows are referred to as service-time-windows. Hence, if a transshipment at a customer of type-1 is necessary, the time window can be neglected but only for the transshipment task. In other words, if the type-1 customer (which is used as VD) is also served by the small or large vehicle, then the time window must be considered for the service task. The service task can be performed before or after the transshipment task.

In summary, the degree of freedom for the LFVRPTW is restricted if time windows are used. To be able to move a tour forwards or backwards in time to make a linkage possible, strongly depends (yet not solely) on the characteristic of the individual time windows. Hence, we can conclude that if certain characteristics like short travel distances, short service and transshipment times and wide time windows are met, the LFVRPTW algorithm may be able to find a solution with synchronised tours.

4 Heuristic approach

4.1 Formal definition of the LFVRPTW

The LFVRPTW is defined on a directed graph $G = (V, E)$ with a set of nodes (V) including the depot (0) and a set of customers (C) which is further divided into customers of type-1 (set A) and type-2 (set B). With each customer are associated a non-negative demand (d_i), non-negative service time (t_i^S) and service time window $[a_i, b_i]$ with earliest starting (a_i) and latest starting time (b_i). The set of edges $(e_{i,j})$ is represented by (E) and they correspond to the travel links between two customers (i) and (j). Each edge has the non-negative travel time $(t_{i,j})$ and non-negative travel cost $(c_{i,j})$. All customers are served by a heterogeneous fleet F of small (F_{SV}) and large (F_{LV}) vehicles (k) with a max. vehicle capacity (Q^k).

All vehicles are dispatched from one physical depot (PD) and the total tour duration must not exceed a given time limit. Customers of type-1 are assumed to have enough parking space and can therefore be visited by both vehicle classes whereas customers of type-2 can only be visited by the small vehicle class. Furthermore, the service for all customers must start within a certain time window. What sets the LVFRPTW apart from other VRP variants is the possibility for the small vehicle class to use the large vehicle class as virtual depot (VD). Instead of driving back to the depot if the capacity is depleted, the small vehicle can meet with a large vehicle at a customer of type-1 for a reloading operation. If goods are transferred from one vehicle to the other, a non-negative transshipment time of (t_{VD}^{TS}) for a transshipment load of (q_{VD}^{TS}) is required.

The detailed mathematical model formulation for the LFVRP(TW) together with some numerical results was already presented in Brandstätter and Reimann (2018b). As the LFVRP requires time related constraints to ensure synchronisation between vehicles, the mathematical model formulation of the LFVRP and LFVRPTW differ only in two constraints (see constraints 44 and 45 in Brandstätter and Reimann 2018b), which secure time window feasibility.

4.2 LFVRPTW algorithm for the *linkage-approach*

The basic concept of the LFVRPTW algorithm is very similar to the LFVRP algorithm we proposed in Brandstätter and Reimann (2018a). To be able to handle time windows (for customer service only), we have to adjust the four improvement strategies *metaheuristic* (ME), *matheuristic* (MA), generating *multiple solutions* (MS) and *local search* (LS) accordingly. Thus, we provide a general overview of the algorithm and describe the adjustments to the respective improvement strategies afterwards.

The overall concept of the algorithm can be described in four steps. In the first step, we solve the VRPTW for all type-2 customers with the metaheuristic Ant Colony Optimisation (ACO). For a general overview of ACO the interested reader is advised to see Dorigo and Blum (2005). The basic concept of ACO is to simulate the behaviour of real-world ants during their search for food. Real-world ants provide a pheromone trail when they search for food. The level of pheromone will increase if another ant uses the path and decrease otherwise. An ant is more likely to choose a path with a higher pheromone level as it was already used by fellow ants indicating a shorter path. Hence, the pheromone level evolves/changes over time and therefore ACO can be assigned to the learning mechanism category of metaheuristics. For our algorithm we use the ACO implementation for the VRPTW of Senarclens de Grancy (2015).

After solving the VRPTW for all type-2 customers, we optimise each small vehicle route individually by applying the principle of matheuristics in our second step. As the name indicates, matheuristics use the benefits of mathematical problem solving and heuristics - thus a hybrid algorithm. Specifically, we use the decomposition technique which extracts a smaller sub-problem from the main problem. In the next step we solve that sub-problem to optimality and reintegrate the solution of the sub-problem back to the main problem. In our case, the sub-problems are represented by the individual small vehicle tours. In other words, we have to solve the Travelling Salesman Problem with Time Windows (TSPTW) for each small vehicle tour. In our algorithm, we use the mathematical problem solver Gurobi Optimizer (see Gurobi Optimization 2016) for the matheuristic strategy.

In the third step, we generate multiple solutions with different characteristics and strategies. The idea of a multiple solution strategy originates from the fact that synchronised tours can hardly be optimised with the well-known VRP operators (e.g. exchange or relocate) often used by local search procedures. Hence, generating multiple solutions and choosing the best among them is a quite promising approach as we already showed in Brandstätter and Reimann (2018a). Therefore, we check the synchronisation against the first- and second-best VD and PD in terms of minimum distance. In addition, we apply the basic concepts of the most common bin packing strategies (e.g. Delorme et al. 2016) named *First Best Fit* (FBF), *First Fit* (FF), *Best Fit* (BF) and *Worst Fit* (WF) together with the originally proposed algorithm of the *Linkage-Approach* from our first paper (see Brandstätter and Reimann 2018b).

From the set of multiple solutions generated in the previous step, we select the best solution with minimum costs and apply a local search procedure in the fourth and final step. The LS procedure searches the solution space (neighbourhood) to find a superior solution. In our example, we use a destroy & repair mechanism for the LS procedure. Finally, the overall algorithm for the LFVRPTW algorithm is presented as pseudo-code in Algorithm 1.

Algorithm 1: LFVRPTW Algorithm

```
apply metaheuristic (ACO) to solve VRPTW for all type-2 customers.// Metaheuristic Strategy
apply matheuristic to optimise all SV tours.// Matheuristic Strategy
for all multiple solution strategies do // Multiple Solutions Strategy
    for the $1^{st}$ and $2^{nd}$ best VD or PD do
        while SV tours can be linked through a VD/PD do
            if linkage is feasible then link tours with a VD/PD.
            else continue
        end
        save solution for type-2 customers temporarily.
    end
end
for all solutions from the previous step do // complete solutions from MS strategy
    sort all VDs in chronological order and create required LV tour(s).
    insert all remaining type-1 customers into SV or LV tours.
    apply matheuristics to optimise all LV tours without synchronization.
    save solution temporarily.
end
choose best solution with minimum cost.
for all synchronised SV tours do // Local Search Strategy
    free all VDs from the selected SV and associated LV tour(s).
    destroy associated LV tour(s) and mark the customers as unserviced.
    apply matheuristics to generate a single SV tour (neglect capacity constraint) for all unserviced
    type-2 customers.
    for all possible VD or PD positions to find a feasible SV tour do
        for all feasible solutions found do
            sort all VDs in chronological order and create required LV tour(s).
            insert all remaining type-1 customers into SV or LV tours.
            save solution after LS temporarily.
        end
    end
    search among all found solutions by the LS procedure.
    if a superior solution is found then proceed with new best found solution.
    else continue
end
```

4.3 Improvement strategy adjustments for time windows

4.3.1 Metaheuristic strategy (ME)

The metaheuristic strategy uses ACO to solve the VRPTW for all type-2 customers. The foundation of ACO is the well-known I1 heuristic proposed by Solomon (1987) which works as follows: In the first step, we select a seed customer and insert it between the start- and end-depot resulting in the first vehicle tour. Afterwards, we add the customers to that tour according to their insertion costs. That procedure continues as long as an insertion results in a feasible tour; feasibility in terms of capacity, time windows and tour duration. If no further customer can be inserted, the algorithm starts a new tour with another seed customer and proceeds with the remaining customers as mentioned before. These steps are repeated until all customers are served.

The ACO metaheuristic constructs a feasible solution for every artificial ant (m) multiple times until a certain stopping criterion (max. number of iterations or time limit) is met. However, contrary to Solomon's I1 heuristic, ACO uses a probabilistic customer selection with the probability $P_{jkl} = \frac{\kappa_{jkl}}{\sum_{i \in I} \kappa_i}$ to insert a customer (k) with the attractiveness (κ_{jkl}) between nodes (j) and (l). Whereas the attractiveness for the seed customer is calculated as $\kappa_{jkl} = c_{jk}(\tau_{jk} + \tau_{kl})$ with (c_{jk}) as distance between depot (j) and customer (k) and ($\tau_{jk} + \tau_{kl}$) as the pheromone levels of the respective edges. The attractiveness to insert a customer (k) into an already existing tour is calculated as the inverse of the normalized costs $\overline{cost_{jkl}}$ (normalised to avoid a division by 0). The costs are calculated as $cost_{jkl} = \gamma_{jkl} T_{jkl}$ if $\gamma_{jkl} \le 0$ and $cost_{jkl} = \gamma_{jkl}/T_{jkl}$ otherwise. Here $\gamma_{jkl} = c_{jk} + c_{kl} - c_{jl} - 2c_{0k}$ represents the additional distance costs and $T_{jkl} = \frac{\tau_{jk} + \tau_{kl}}{2\tau_{jl}}$ the pheromone trail value.

As mentioned before, the ants lay a pheromone trail on their used paths. That pheromone level decreases (evaporates) over time if the path is not used. On the other hand, if that path is used by another ant, the pheromone level increases. Therefore, before the first iteration starts, the pheromone level for all edges is set to 1 ($\tau_{ij} = 1$) and updated after each iteration by $\tau_{ij} = \tau_{ij}\rho + (1 - \rho)x_{ij}$ with (ρ) as pheromone persistence factor and x_{ij} as decision variable if the path is in the current solution. Finally, the ACO implementation is concluded by a LS procedure with three operators (shake-reduce, exchange and relocate). A detailed description of the ACO design can be found in Senarclens de Grancy (2015).

4.3.2 Matheuristic strategy (MA)

In Brandstätter and Reimann (2018a) we use the mathematical model formulation of Dantzig et al. (1954) to solve the Travelling Salesman Problem (TSP). Yet, this formulation does not include any time restrictions. Therefore, we use the TSPTW formulation of Dash et al. (2012) which was originally proposed as Big-M-Formulation by Ascheuer et al. (2001). However, this formulation does not fully cover our needs and some additional adjustments are necessary. The final TSPTW model formulation for our algorithm is as follows.

The set of nodes (V) is represented by the set of customers (C) and the starting (p) and end (q) depot (for simplification we split between start and end depot). All nodes have associated coordinates, service times (t^s) and a time window with the earliest starting time (a_i) and latest starting time (b_i). The driving distance ($c_{i,j}$) between respective nodes (i) and (j) is represented by the Euclidean distance. Finally, the objective is to minimise the total costs (z).

Parameters

z	total costs
C	set of customers
p	start depot
q	end depot
$V = C \cup \{p, q\}$	set of nodes
$c_{i,j}$	distance between node i and j

c^f	fuel cost [currency/km]
c^w	drivers wage [currency/hour]
t_i^a	arrival time at node i
$t_{i,j}^d$	driving time from node i to j
t_i^s	service time of node i
a_i	earliest starting time of node i
b_i	latest starting time of node i
M	sufficient large number

Decision variable

$$x_{i,j} = \begin{cases} 1, & \text{if vehicle uses edge } i \text{ to } j \\ 0, & \text{else} \end{cases}$$

The objective function considers the total driven distance ($\sum_{i \neq j,(i,j) \in V} c_{i,j} x_{i,j}$) and the total required time ($t_q^a - t_p^a$). The cost factors (c^f) represent the fuel consumption of the vehicle and (c^f) the drivers wage [currency/hour]. The objective function is presented in Eq. (4).

Objective function

$$\min z = c^f * \sum_{i \neq j,(i,j) \in V} c_{i,j} x_{i,j} + c^w * (t_q^a - t_p^a) \tag{4}$$

Constraints

$$\sum_{j \in \{C \cup \{q\}\}} x_{i,j} = 1 \qquad i \neq j, \quad \forall i \in C \cup \{p\} \tag{5}$$

$$\sum_{i \in \{C \cup \{p\}\}} x_{i,h} - \sum_{j \in \{C \cup \{q\}\}} x_{h,j} = 0 \qquad i \neq h, \quad h \neq j, \quad \forall h \in C \tag{6}$$

$$t_i + t_{i,j}^d + t_i^s - M * (1 - x_{i,j}) \leq t_j \qquad i \neq j, \quad \forall (i,j) \in V \tag{7}$$

$$t_i^a \geq a_i x_{i,j} \qquad i \neq j, \quad \forall (i,j) \in V \tag{8}$$

$$t_i^a \leq b_i + M * (1 - x_{i,j}) \qquad i \neq j, \quad \forall (i,j) \in V \tag{9}$$

$$x_{i,j} \in \{0, 1\} \qquad i \neq j, \quad \forall (i,j) \in V \tag{10}$$

$$M = max(b_i)_{\forall i \in V} + \phi \qquad \phi > 0 \tag{11}$$

Constraints (5) ensure that all customers are visited, whereas the continuous routing flow is assured with (6). The time progress is assured with constraints (7). The compliance with time windows is secured with constraints (8)–(9). Finally, the last two constraints secure the definition of the decision variable (10) and that M is sufficiently large (11). The validity of the model has been tested against the formulation of Dash et al. (2012).

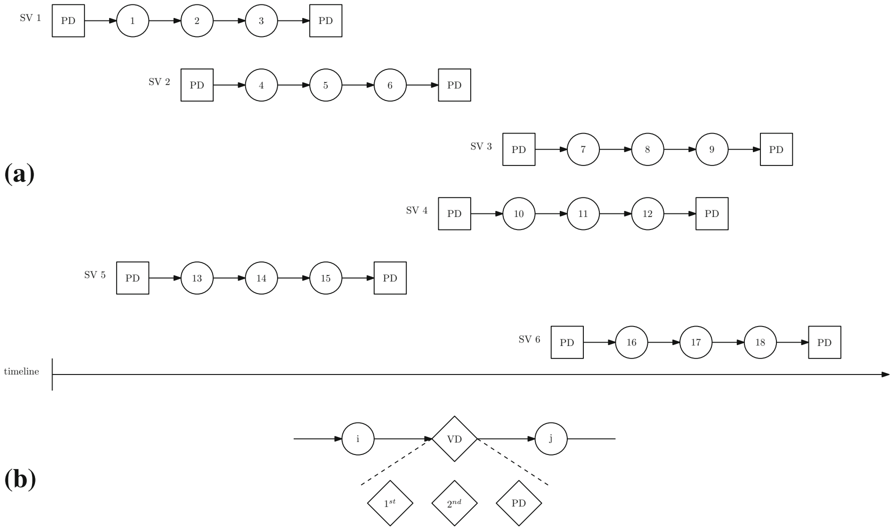


Fig. 4 Multiple solutions strategy

4.3.3 Multiple solutions strategy (MS)

The multiple solution strategy is the main part of our algorithm. For a better understanding, we describe the four previously mentioned bin packing algorithms briefly and also provide a small explanatory example in Fig. 4. Figure 4a shows the result after the MA strategy with six SV tours already positioned according to their time windows. What we can observe is that a linkage might only be possible between SV1 → {SV3, SV4, SV6}, SV2 → {SV3, SV4, SV6} and SV5 → {SV3, SV6}. To begin with, the FBF algorithm selects the first SV tour as starting tour and checks if a linkage is possible with the next tour. Thus, we replace the edge from the last customer of the first tour to the PD ($e_{i,PD}$) by the edge to the VD ($e_{i,VD}$) and the edge from the PD to the first customer of the next tour ($e_{PD,j}$) with the edge from the VD to the first customer ($e_{VD,j}$), respectively (see Fig. 4b: within the MS strategy, we check the best VD location for the first- and second-best customer as well as the PD. However, for simplification of this example, we consider only the best VD for now). If the linkage is feasible, it is stored temporarily and the next possible SV tour is checked. That procedure continues until all other possible SV tours are tested. After the linkage between the starting tour and all other tours is checked, the best possible linkage (if there is any) is selected - therefore *First Best Fit*.

The other algorithms are based on the same principle: selecting an SV tour as a starting tour and checking a possible linkage with the other possible SV tours. They only differ in the result selection when a linkage is accepted. For instance, the FF algorithm accepts a linkage as soon as a feasible linkage is found. Whereas the result selection in the BF/WF algorithm is not performed until every SV tour has been used as starting tour to select the best/worst result in terms of minimum resulting cost, the BF/WF algorithm is repeated until no further linkage is possible.

The multiple solution strategy developed for the LFVRP can almost be used for the LFVRPTW, with only two exceptions. First, if we do not use time windows, a small vehicle tour can be considered forwards (FW: e.g. $PD \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow PD$) and backwards (BW: e.g. $PD \rightarrow 4 \rightarrow 3 \rightarrow 2 \rightarrow 1 \rightarrow PD$). In other words, if we try to link two SV tours through a VD, we are able to create 4 possible combinations (FW-FW, FW-BW, BW-FW and BW-BW). As the VD will be inserted between the last customer of the first tour and the first customer of the second tour, it is possible that a different VD is used which may result in lesser costs. Yet, with time windows in place, the tour direction is predefined through the time windows and therefore we cannot consider the backwards version of the tour in the LFVRPTW.

The second change required for the LFVRPTW is the feasibility check. In the LFVRP algorithm we only had to assure that the constraints for capacity, transshipment (small and large vehicle have to be at the same place at the same time to perform a transshipment) and maximum allowed tour duration are met. Now we also have to ensure that all service starting times are aligned with their respective customer time windows. For a better understanding, we illustrate the feasibility check with two simplified equations [see Eqs. (12) and (13)].

$$q_j^a = \begin{cases} q_i^a - d_i\alpha + q_{SV}^{TS}\beta, & \text{if} \quad q_j^a \geq d_j \\ \text{not feasible,} & q_j^a < d_j \end{cases}$$
$$i \neq j, \quad \forall (i, j) \in V, \quad \alpha \in \{0, 1\}, \quad \beta \in \{0, 1\} \tag{12}$$

$$t_j^a = \begin{cases} t_i^a + t_i^s\alpha + t^{TS}\beta + t_{i,j}^d, & \text{if} \quad a_j \leq t_j^a \leq b_j \\ a_j, & t_j^a < a_j \\ \text{not feasible,} & t_j^a > b_j \end{cases}$$
$$i \neq j, \quad \forall (i, j) \in V, \quad \alpha \in \{0, 1\}, \quad \beta \in \{0, 1\} \tag{13}$$

The feasibility check for the capacity is presented in Eq. (12). Here (q_j^a) represents the load of the vehicle upon arrival and is calculated as the vehicle load upon arrival at the previous customer (q_i^a) minus the customer demand ($d_i\alpha$: if a service is performed ($\alpha = 1$)) plus the transshipment load ($q_{SV}^{TS}\beta$: if a transshipment is performed ($\beta = 1$)). If it ever occurs that ($q_j^a < d_j$), then the SV tour is infeasible. The feasibility check for the time is presented in Eq. (13). The arrival time at customer (j) is represented by (t_j^a) and is determined as the arrival time (t_i^a) at the previous customer (i) plus the customer service time ($t_i^s\alpha$: if a service is performed ($\alpha = 1$)) plus the transshipment time ($t^{TS}\beta$: if a transshipment is performed ($\beta = 1$)). However, if the arrival time is less than the earliest starting time ($t_j^a < a_j$), then the vehicle must wait until the time window opens. Therefore, the arrival time will be set to the value of the earliest starting time. If the arrival time is larger than the latest starting time ($t_j^a > b_j$), then the SV tour is infeasible and the feasibility check failed.

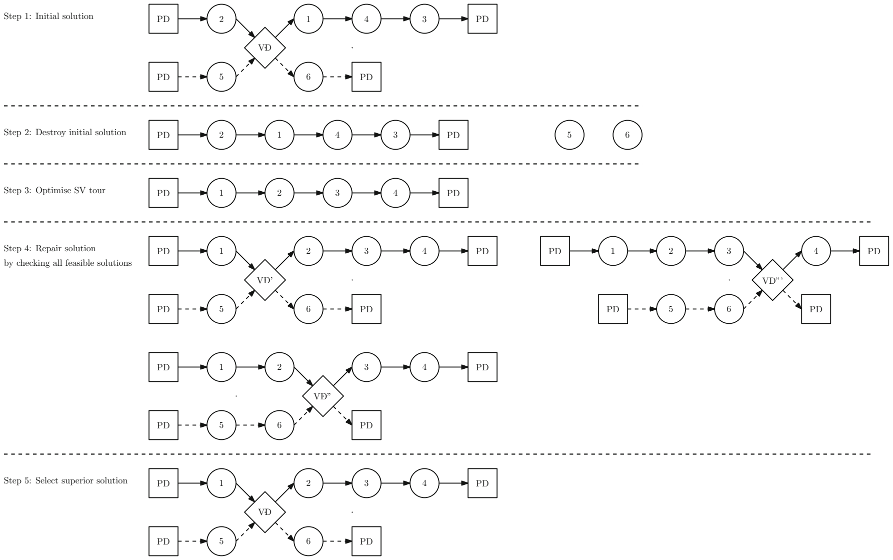


Fig. 5 Local search procedure

4.3.4 Local search strategy (LS)

For the last improvement step we use a large neighbourhood search (LNS) to further improve our solution. The basic principle is to search for a superior solution in the neighbourhood of a given solution. There are multiple LNS heuristics but we decided to use a destroy-and-repair procedure. That procedure destroys the initial solution and tries to repair it afterwards by using other characteristics than before. Our destroy-and-repair procedure requires five steps and works as follows.

We start by checking all SV tours for synchronisation as the LS procedure will only be applied to synchronised tours. Thus, the first SV tour with synchronisation is selected. In the next step, all reload positions (VDs and PD returns) from the SV tour and all reload positions and customers from the associated LV tour(s) are freed. Once the initial solution is destroyed, we optimise the SV tour with the MA strategy. Afterwards we insert all required VDs or PD visits back into the SV tour. But now we check all possible insertions that lead to a feasible solution. Thus, if we find a feasible solution, the associated LV tour(s) is (are) constructed (see Algorithm 1 lines 12–17) and stored temporarily. In the fifth and final step, we compare the initial solution with all temporarily stored solutions and select the superior one. Afterwards we apply the LS procedure to all other synchronised SV tours.

For a better understanding, we provide a small illustrative example in Fig. 5. The initial solution for the LS procedure consists of 1 SV tour ($PD \rightarrow 2 \rightarrow VD \rightarrow 1 \rightarrow 4 \rightarrow 3 \rightarrow PD$) with 4 customers and one VD and one LV ($PD \rightarrow 5 \rightarrow VD \rightarrow 6 \rightarrow PD$) tour with two customers. In the second step the VD is freed from the SV tour and the LV tour is destroyed completely. Afterwards the SV tour is optimized which results in a new SV tour $PD \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow PD$. In the fourth step all possible insertions are checked and stored temporarily. In our example the first solution will be accepted as the new solution and the LS procedure continues.

5 Computational results

For our computational results we used the following hard- and software configuration. The coding was done in programming language C and compiled with GNU GCC Compiler on a 4 * Intel® Core$^{\mathrm{TM}}$ i7-5557U CPU @ 3.1 GHz processor and 16 GB DDR3 RAM (1.6 MHz) under a 64-Bit Operating System (Kubuntu 14.04). For the mathematical problem solver, we used the optimization software Gurobi Optimizer (Version 6.5.1) and coded the mathematical formulation of the TSPTW with the programming language Python (Version 2.7.6).

Furthermore, our analysis is divided into two parts. In the first part we want to understand the impact of different time window characteristics. To do so, we decided to generate multiple instances with large, medium and tight time windows and compare our results to the classical VRPTW and HFVRPTW. For the second part we will compare our results to the best-known results (BKS) of the LFVRPTW. However, a thorough comparison is not possible because some vital information is missing on how the BKS were obtained. Yet, for the sake of completeness, we decided to make logical assumptions to be able to compare our results to the BKS.

5.1 Setup for the LFVRPTW algorithm

The algorithm for the LFVRPTW was programmed to be as flexible as possible. If not indicated otherwise, the algorithmic set-up is as follows. The input files of the test instances consist of a single depot (PD) and multiple customers with associated characteristics (x-coordinate, y-coordinate, demand, time window: earliest starting time and latest starting time, service time and customer type). The driving distance between two nodes is the Euclidean distance and driving time the Euclidean distance divided by the average speed of the vehicle class. Although the service time is given by the test instance, we designed the algorithm to use unit-dependent service and transshipment times. To achieve that, we defined the required time for a complete transshipment of the small vehicle's capacity and calculated the transshipment and service time according to transshipment load and demand. As shown in our second paper we remained with 10 repetitions per instance as default. If not explicitly indicated, the results are the best found results out of these 10 repetitions.

Furthermore, all four improvement strategies (ME, MA, MS and LS) are enabled by default. Although we were able to solve all sub-problems of the matheuristic strategy in Brandstätter and Reimann (2018a) in a reasonable amount of time (less than 3 s), we realized some difficulties when we applied that principle to the LFVRPTW. The reasons for that are the additional time constraints used in the TSPTW model formulation. We figured out that larger sub-problems required more than 60 s to find a feasible/optimal solution. As a result, we decided to set the time limit for the calculation time of the mathematical problem solver to 10 s. The solver will only return a solution to the algorithm if a superior solution is found or otherwise the algorithm will proceed with the starting solution.

Finally, the main objective of our comparison are the total required costs which are calculated as the sum of (i) fixed costs (number of used vehicles $no.^k$ multiplied by the associated fixed costs f^k_{fix}), (ii) fuel consumption costs (total distance $dist.^k$ per vehicle class k multiplied by the variable costs per length unit f^k_{fuel}) and (iii) costs for the drivers wage (total required time $time^k$ of the vehicle class k multiplied by the variable costs per time unit f^k_{wage}).

$$\begin{aligned} costs = no.^{SV} * f^{SV}_{fix} + dist.^{SV} * f^{SV}_{fuel} + time^{SV} * f^{SV}_{wage} \\ + no.^{LV} * f^{LV}_{fix} + dist.^{LV} * f^{LV}_{fuel} + time^{LV} * f^{LV}_{wage} \end{aligned} \tag{14}$$

5.2 Impact of different time window characteristics

Since time windows have a huge impact on the effectiveness of the routing (see Russo and Comi 2010), a thorough analysis is essential. Therefore, we decided to analyse three time window categories: 2 h ([0–2],[2–4],[4–6],[6–8],[8–10] and [10–12]), 4 h ([0–4],[4–8] and [8–12]) and 6 h ([0–6] and [6–12]) with a tour duration limit (and maximum time window of the depot) of 12 h. As for the customer location and demand values, we chose the C, R and RC sets of the well-known Solomon problem instances (see Solomon 1987). However, as the customer location will only differ between the C, R and RC instance sets, the customer distribution of type-1 and type-2 customers was initially set and not changed afterwards. We assumed that the majority of the customers are hardly accessible and therefore decided to set the customer distribution to 1:3. Thus, 25% of the customers are of type-1 and can be chosen as a VD, whereas the majority of the customers (75%) are of type-2. To gain a large set of different test instances, we created 10 test instances with randomly assigned time windows for each test instance set and time window category. Consequently, we generated 90 (3[C,R,RC]*3[2, 4, 6]*10 = 90) test instances for our analysis.

Moreover, we used the same costs and service/transshipment setup as in Brandstätter and Reimann (2018a). We assumed a reload time of 15 min for the full capacity of the small vehicle and the service times in relation to the demand of each customer. For the large vehicle we consider a truck with a trailer (40t payload—capacity for the test instances is the sum of the total customer demand) with daily costs of €200 per vehicle (f^{LV}_{fix}) , fuel consumption of €0.35/km (f^{LV}_{fuel}—35l per 100 km) and a drivers salary of €30 per hour (f^{LV}_{wage}) and for the small vehicle we use a cargo-bike (150 kg payload—capacity for the test instance set is 50 units) with costs of €2 per day and vehicle (f^{SV}_{fix}), €0.00/km (f^{SV}_{fuel}: no fuel consumption) and the drivers salary of €20 per hour (f^{SV}_{wage}). The average speed of the small vehicle is assumed with 15 km/h and 40 km/h for the large vehicle.

For a better comparison we present the results for the LFVPRTW together with the VRPTW and HFVRPTW. The VRPTW results were obtained by changing the customer type for all customers to type-2 and the HFVRPTW results by disabling the reload option at a VD. To get a deeper insight into the analysis with time windows we decided to present three aggregated tables. Table 1 shows the aggregated results of the different problems together with the time window category and instance set classification. The sole time window comparison regardless of the instance set is presented in Table 2. Besides the total costs, we also present the number of vehicles, driven distance, used time, number of reloads and required CPU time. Our analysis will be concluded by a deviation analysis and statistical comparison in Table 3. In all three tables the best results are indicated by bold values and the minimum results for all three problems and for all generated instances can be found in the appendix.

We observe that the LFVRPTW is the superior approach for all time window categories and almost all instance sets. Only for the mixed customer distribution with the 4 h time window category the HFVRPTW was able to provide a superior result. For the randomly generated customer distribution within the same time window category, the HF- and LFVRPTW provide almost the same results. As expected, the VRPTW is the least effective approach. That is due to the fact that the other two approaches allow the use of large vehicles. If large vehicles are allowed, the HF- and LFVRPTW are able to reduce the number of required small vehicles significantly (between 10 and 15 vehicles).

Another interesting observation is that, on average, the required CPU time for the LFVRPTW is higher than for the other two approaches. Yet, the difference between the results of the HFVRPTW and the LFVRPTW is rather small compared to the VRPTW. The reasons for that can be found within the two improvement strategies MS and LS. As the usage of a VD is prohibited within the VRPTW and HFVRPTW, the MS and LS strategy will not be used. Furthermore, the LFVRPTW requires significantly more CPU time for the 6 h time window category (compare CPU times in Table 2), whereas the CPU time for the VRPTW and HFVRPTW show hardly any deviation regardless of the time window category or instance set. If time windows are tight, the possibilities for suitable VD locations or potential linkage of tours are limited. However, these possibilities increase if time windows become larger. Consequently, more possibilities within the MS and LS strategies need to be considered. As a result, additional computation time is required and superior final results are found.

Finally, we present the average costs per time window category together with the relative percentage deviation (RPD) and the p-values of the statistical comparison in Table 3. The RPD is calculated as $RPD = 100 * \frac{Cost_x - Cost_{LFVRPTW}}{Cost_x}$, where $x \in \{VRPTW, HFVRPTW\}$. As for the statistical comparison, we assume an equal solution quality with the null-hypothesis H_0 and the alternative-hypothesis H_1 if the difference in solution quality is significant. If the p-value is less than 0.05, the null-hypothesis can be rejected. For our statistical analysis we used the Wilcoxon-Signed-Rank-Test (WSRT see Wilcoxon 1945) which we performed with R.

Overall, the results of the LFVRPTW are superior compared to those of the VRPTW and HFVRPTW. However, when we have a closer look at the different time window categories, the HFVRPTW can compete with the LFVRPTW if the time windows are of medium size (4 h). The reason for that are the required number of vehicles. While the HFVRPTW requires the same number of large vehicles if the time windows are medium or large, more large vehicles are needed if the time windows are tight. Although this may be true for the HFVRPTW, it is not the case for the LFVRPTW. If time windows are medium or tight, the LFVRPTW requires almost the same number

Table 1 Instance set analysis

Problem[a]	TW cat.[b]	Inst.-Set	Cost	SV	LV	Dist SV	Time SV	Dist LV	Time LV	Reload	CPU[c]
HFVRPTW	2	c	9084	26.0	1.6	3960.26	15331.79	678.41	1078.31	0.0	6.004
		r	6889	23.0	1.0	2901.38	11707.81	520.36	809.44	0.0	5.997
		rc	8913	28.0	1.6	4086.12	14904.90	710.59	1114.48	0.0	5.998
	4	c	7604	26.0	1.0	3948.03	13094.52	605.54	790.86	0.0	5.999
		r	5799	23.0	1.0	2882.90	9805.45	422.09	669.57	0.0	5.997
		rc	**7494**	28.0	1.0	4070.11	12817.49	648.77	803.42	0.0	5.998
	6	c	6149	26.0	1.0	3930.74	10598.98	511.80	557.70	0.0	5.928
		r	4541	23.0	1.0	2849.41	7717.44	371.03	409.03	0.0	6.002
		rc	6326	28.0	1.0	4042.84	10920.91	528.78	565.78	0.0	5.991
LFVRPTW	2	c	**8681**	19.3	2.3	3173.52	13771.83	1005.05	1259.18	6.8	4.188
		r	**6773**	17.8	1.2	2700.80	11353.29	564.39	831.07	5.3	3.922
		rc	**8776**	21.2	2.3	3577.53	13941.64	983.20	1278.85	6.7	4.029
	4	c	**7457**	19.6	2.0	3250.02	11915.21	845.01	1019.50	6.5	5.194
		r	**5795**	16.6	1.5	2635.80	9422.21	541.07	748.68	6.3	4.319
		rc	7679	21.2	2.5	3557.37	12063.83	879.26	1062.83	6.6	4.383
	6	c	**5744**	21.5	1.6	3378.53	9110.32	671.59	787.49	4.9	9.659
		r	**4287**	18.6	1.1	2587.31	6992.86	404.03	523.25	4.7	13.309
		rc	**6023**	23.2	1.5	3630.94	9776.66	634.28	755.13	4.9	7.712
VRPTW	2	c	10150	37.0	0.0	5338.60	20151.99	0.00	0.00	0.0	3.040
		r	7965	30.0	0.0	3761.23	15809.49	0.00	0.00	0.0	3.027
		rc	9509	35.0	0.0	5155.88	18877.43	0.00	0.00	0.0	3.002
	4	c	8736	37.0	0.0	5338.04	17324.37	0.00	0.00	0.0	3.010

Table 1 continued

Problem[a]	TW cat.[b]	Inst.-Set	Cost	SV	LV	Dist SV	Time SV	Dist LV	Time LV	Reload	CPU[c]
	6	r	6562	30.0	0.0	3722.73	13003.62	0.00	0.00	0.0	2.987
		rc	8288	35.0	0.0	5107.78	16436.11	0.00	0.00	0.0	2.996
		c	7243	37.0	0.0	5315.31	14337.06	0.00	0.00	0.0	2.949
		r	5039	30.0	0.0	3675.23	9957.62	0.00	0.00	0.0	3.014
		rc	6873	35.0	0.0	5035.78	13605.74	0.00	0.00	0.0	3.004

[a] Problems for comparison: VRPTW (all customers of type-2), HFVRPTW (2 vehicle classes but no transshipment allowed) and LFVRPTW.
[b] Time windows of 2, 4 and 6 h.
[c] CPU time in seconds

Table 2 Time window analysis

TW cat.[a]	Problem[b]	Cost	SV	LV	Dist SV	Time SV	Dist LV	Time LV	Reload	CPU[c]
2	HFVRPTW	8295	25.7	1.4	3649.25	13981.50	636.45	1000.74	0.0	6.000
	LFVRPTW	**8077**	19.4	1.9	3150.62	13022.25	850.88	1123.03	6.3	4.047
	VRPTW	9208	34.0	0.0	4751.90	18279.64	0.00	0.00	0.0	3.023
4	HFVRPTW	**6966**	25.7	1.0	3633.68	11905.82	558.80	754.62	0.0	5.998
	LFVRPTW	6977	19.1	2.0	3147.73	11133.75	755.11	943.67	6.5	4.632
	VRPTW	7862	34.0	0.0	4722.85	15588.03	0.00	0.00	0.0	2.998
6	HFVRPTW	5672	25.7	1.0	3607.66	9745.77	470.54	510.84	0.0	5.974
	LFVRPTW	**5351**	21.1	1.4	3198.93	8626.61	569.97	688.62	4.8	10.227
	VRPTW	6385	34.0	0.0	4675.44	12633.47	0.00	0.00	0.0	2.989
Average	HFVRPTW	6978	25.7	1.1	3630.20	11877.70	555.26	755.40	0.0	5.991
	LFVRPTW	**6802**	19.9	1.8	3165.76	10927.54	725.32	918.44	5.9	6.302
	VRPTW	7818	34.0	0.0	4716.73	15500.38	0.00	0.00	0.0	3.003

[a] Time window category of 2, 4 and 6 h.

[b] Problems for comparison: VRPTW (all customers of type-2), HFVRPTW (2 vehicle classes but no transshipment allowed) and LFVRPTW.

[c] CPU time in seconds

of small and large vehicles. Only if the time windows are large, the LFVRPTW is also able to reduce the number of large vehicles. Having that in mind and that the costs for the large vehicle class are a lot higher than the costs for the small vehicle class, the difference in the number of large vehicles has a substantial effect on the total costs. This can be seen in Table 3 with the RPD analysis. With tight and large time windows the LFVRPTW provides superior results. Only if the time windows are of medium size, the HFVRPTW provides better results. From a statistical point of view, the above observation can be confirmed for tight (instance set c) and large sized time windows with a significant advantage of the LFVRPTW. However, for medium sized time windows, the LFVRPTW and HFVRPTW provide equal results in terms of solution quality.

5.3 Comparison to best-known LFVRPTW results

To the best of our knowledge, the current BKS for the LFVRPTW have been provided by Chen and Wang (2012). To be able to compare our results we had to change the set-up of our algorithm slightly to the information provided. Nevertheless, not all required parameter information was provided for a proper comparison. Yet, we were able to find some missing parameters in Chen et al. (2011a) and concerning the rest we made logical assumptions. However, we were still not able to solve all test instances as some customers could not be served within their respective time windows. Thus, we had to adjust some of the given customer time windows. In summary, we collected the parameters from multiple papers, made logical assumptions and adjusted some customer time windows to be able to find feasible solutions. But we were not able to make a solid comparison to the BKS for the LFVRPTW due to the prior mentioned problems. Therefore, we decided to skip the BKS comparison from the main part of this paper. Yet, for the sake of completeness, we presented our comparison approach in the appendix.

6 Conclusion

Time windows play an important role for city logistics because the delivery of goods is very often time dependent (e.g. opening hours or delivery only in the morning hours). Therefore, time windows need to be addressed accordingly in vehicle routing research. In our previous papers we introduced a new problem which is called the LFVRP; where small and large vehicles are used for delivery and synchronisation between vehicles is possible. Yet, time windows have not been considered so far. With this paper we close this gap by providing a thorough analysis on the impact of time windows on the LFVRPTW.

We started with an overview of relevant literature followed by a structural analysis to better understand the problem with time windows. Due to earlier findings, we further developed/extended the *Linkage-Approach* by means of the four improvement strategies *metaheuristic*, *matheuristic*, generating *multiple solutions* and *local search* from our previous paper. The metaheuristic strategy (ME) creates small vehicle tours for all

Table 3 Relative percentage deviation and statistical comparison

TW cat.	Inst.-Set	Problem			VRPTW/LFVRPTW[a]		HFVRPTW/LFVRPTW[a]	
		VRPTW	HFVRPTW	LFVRPTW	RPD[b](%)	p-value[c]	RPD[b] (%)	p-value[c]
2	c	10150	9084	**8681**	14.48	**0.00**	4.44	**0.00**
	r	7965	6889	**6773**	14.96	**0.00**	1.68	0.23
	rc	9509	8913	**8776**	7.70	**0.00**	1.53	0.38
4	c	8736	7604	**7457**	14.64	**0.00**	1.94	0.13
	r	6562	5799	**5795**	11.68	**0.00**	0.06	1.00
	rc	8288	**7494**	7679	7.35	**0.00**	−2.47	0.05
6	c	7243	6149	**5744**	20.69	**0.00**	6.59	**0.00**
	r	5039	4541	**4287**	14.91	**0.00**	5.59	**0.00**
	rc	6873	6326	**6023**	12.36	**0.00**	4.79	**0.00**
Average results 2 h		9208	8295	**8077**	12.28	**0.00**	2.64	**0.00**
Average results 4 h		7862	**6966**	6977	11.25	**0.00**	−0.16	0.89
Average results 6 h		6385	5672	**5351**	16.18	**0.00**	5.65	**0.00**

[a] Comparison of average results.

[b] Relative percentage deviation (RPD).

[c] Statistical results: p-value $< 0.05 \rightarrow H_0$ rejected

type-2 customers while the matheuristic strategy (MA) tries to find the optimal solution for each tour. As the name already indicates, the multiple solution strategy (MS) generates multiple solutions by using different construction techniques and reload positions. Finally, the local search strategy (LS) uses a destroy-and-repair mechanism to further improve the solution. In our computational analysis we developed 90 test instances with different time windows using three different time window categories: tight (2 h), medium (4 h) and large (6 h). For a better comparison, we provided results for the VRPTW and HFVRPTW. In conclusion, the LFVRPTW provides significantly superior results for tight and large time windows. For medium sized time windows, the LFVRPTW requires more large vehicles and therefore the HFVRPTW is slightly (yet not significantly) superior.

The insights of this paper serve as a solid foundation for future research on the LFVRPTW. Although the width of the time windows is crucial for the advantage of the LFVRPTW, alternative algorithmic approaches could provide superior results. For example, it would be interesting to merge the MS with the ME strategy and create multiple starting solutions. In this case, the algorithm would have more possibilities for potential reload positions. Another approach would be to use an alternative construction approach like parallel route construction heuristics. Clearly the LFVRPTW shows great potential—especially for city logistics—but still needs to be investigated in more detail.

Acknowledgements Open access funding provided by University of Graz.

Appendix

According to Chen and Wang (2012) the required parameters are as follows. For the small vehicle class the costs (in New Taiwan Dollar NTD) are 600 NTD/day for rent and insurance (f_{fix}^{SV}), 0,77 NTD/km fuel cost (f_{fuel}^{SV}) and 120 NTD/h the drivers wage (f_{wage}^{SV}), whereas the costs for the large vehicle are 1450 NTD/day for rent and insurance (f_{fix}^{LV}), 3.3 NTD/km fuel cost (f_{fuel}^{LV}) and 270 NTD/h the drivers wage (f_{wage}^{LV}). Furthermore, the small vehicle capacity is associated with each test instance and the large vehicle capacity is ten times the small vehicle capacity.

However, not all required information (e.g. average speed and transshipment/reload time) has been provided by Chen and Wang (2012). Therefore, we assumed the average speed of 40 km/h (as presented for the small vehicle class in Chen et al. 2011a) for both vehicle classes and the transshipment/reload time of 10 units. Yet, with the provided information at hand, it was still impossible to provide a feasible solution for some instances as some customers could not be served within their provided time windows. In these cases the earliest starting time (a_i) plus service time (t_i^s) plus travel time back to the depot (t_{ij}) is greater than the latest arrival time at the depot (b_0): $a_i + t_i^s + t_{ij} > b_0$

or the travel time to the customer (t_{0i}) is greater than the latest starting time of the respective customer (b_i): $t_{0i} > b_i$. As the customer time windows were provided in four categories [08:00–20:00], [08:00–12:00], [12:00–17:00] and [17:00–20:00], we had to change the respective customer time windows to the next possible category to be able to create feasible solutions.

The results of our algorithm are summarized in Table 4 together with the results of the ELFVRPTW provided by Chen and Wang (2012). For 10 out of 17 test instances we were able to provide superior results compared to the ELFVRPTW. Furthermore, in terms of average total costs, our algorithm totally outperformed the ELFVRPTW (13944 compared to 18719). Unfortunately, Chen and Wang (2012) did not provide the number of vehicles nor required distances and time which made it difficult to provide a more detailed analysis. Yet, if we have a closer look at the reload column, we can see that the algorithm was not able to link some small vehicle tours through a VD or the PD. In other words, as no reload or transshipment takes place and no large vehicle is used, the result of our algorithm can be seen as a VRPTW result. Although the computation time is hardly comparable due to different hardware configuration, it is worth mentioning that our algorithm needs on average approx. 4 times longer than the ELFVRPTW. The reason for that is the usage of our four improvement strategies ME, MA, MS and LS.

In summary, our algorithm clearly outperforms the algorithm provided by Chen and Wang (2012), but we require more computation time and do not use a VD or PD for reloading. However, it is difficult to perform a thorough comparison because Chen and Wang (2012) did not provide the necessary information to comprehend the reported results. In addition, we also had to make some assumptions and some instances needed a time window adjustment to be able to create a feasible solution. Hence, for future research on the LFVRPTW, we recommend to use our results of Table 4 which provide all necessary information for a solid comparison.

Data tables

In the following three tables we present the results for the LFVRPTW in Table 5, VRPTW in Table 6 and HFVRPTW in Table 7.

Table 4 Chen: LFVRPTW result comparison

Instance	ELFVRPTW[a]				LFVRPTW					
	Cost	SV[d]	Reload	CPU[c]	Cost	Distance	Time	SV	Reload	CPU[c]
A32	9199	n.a.	0	0.63	**9082**	995.89	1771.56	5	0	2.981
A34	**8365**	n.a.	0	0.35	8663	1013.73	1627.57	5	0	2.741
A38	**9297**	n.a.	0	0.81	9792	1064.04	1790.83	6	0	3.983
A45	**10874**	n.a.	0	1.72	11607	1484.60	2087.98	7	0	8.617
A46	**9934**	n.a.	0	1.27	10043	1204.82	1838.50	6	0	4.697
A60[b]	**13919**	n.a.	1	2.61	15307	1643.61	2880.62	9	0	8.315
A61	**11889**	n.a.	1	1.82	15530	1402.10	2816.80	10	0	4.966
A64	14810	n.a.	1	2.97	**14521**	1786.53	2581.90	9	0	8.844
A65	14626	n.a.	1	2.94	**14110**	1555.86	2703.98	8	0	8.779
A69	14800	n.a.	0	1.64	**13822**	1586.44	2600.21	8	0	20.146
A80[b]	20753	n.a.	2	5.39	**17831**	2229.33	3371.49	10	0	25.416
C51	6506	n.a.	0	1.58	**6196**	744.49	1274.33	3	0	32.992
C81[b]	30718	n.a.	0	6.00	**18554**	2725.59	3685.24	9	0	24.332
C161[b]	105802	n.a.	0	45.45	**42779**	6572.56	8172.56	22	0	97.386
F45	10145	n.a.	0	1.16	**8707**	1037.52	1635.91	5	0	15.946
F72	**5511**	n.a.	2	4.19	6432	387.35	1244.50	4	0	37.918
F135[b]	21076	n.a.	0	21.08	**14067**	1539.57	2893.85	7	0	58.415
Average	18719	n.a.	0.47	5.98	13944	1704.35	2645.75	7.8	0	21.557

The bold numbers indicate the superior approach in terms of minimum costs

[a] Results from Chen and Wang (2012).

[b] Time windows adjustment necessary to gain a feasible solution.

[c] Due to different hardware configuration CPU times are not comparable.

[d] No. of small vehicles are not provided in Chen and Wang (2012)

Table 5 Results LFVRPTW

Instance[a]	Cost	Dist.	Time	No. SV	No LV	Dist SV	Time SV	Dist LV	Time LV	Reload	CPU[b]
c-2-01	8698	4145.40	15078.11	18	2	3061.06	13705.93	1084.34	1372.18	8	4.274
c-2-02	8496	4280.33	14799.78	20	2	3281.33	13575.24	999.00	1224.54	6	4.026
c-2-03	8693	4402.30	14531.10	18	3	3199.82	13048.03	1202.48	1483.08	8	4.283
c-2-04	8679	4122.93	14810.51	20	3	3186.89	13584.41	936.05	1226.10	7	4.123
c-2-05	8681	4158.93	15324.83	20	2	3261.12	14268.50	897.82	1056.33	6	4.061
c-2-06	9100	4130.53	16083.87	20	2	3182.39	14938.86	948.13	1145.01	6	5.013
c-2-07	8726	4074.70	15289.58	20	2	3137.66	14034.84	937.04	1254.74	6	3.732
c-2-08	8691	4213.31	14790.60	19	3	3236.27	13528.39	977.04	1262.21	7	4.162
c-2-09	8609	4161.83	14926.04	19	2	3083.54	13602.41	1078.28	1323.63	7	4.047
c-2-10	8433	4095.47	14675.69	19	2	3105.13	13431.68	990.34	1244.01	7	4.161
c-4-01	7266	4050.98	13173.24	21	1	3380.09	12364.63	670.89	808.61	5	4.174
c-4-02	7493	4263.74	13481.52	22	1	3454.09	12580.55	809.65	900.97	4	3.394
c-4-03	8006	4202.80	14674.33	23	1	3563.74	13876.64	639.07	797.68	3	4.145
c-4-04	7445	3909.41	12020.28	17	4	3018.89	10861.71	890.51	1158.57	9	4.226
c-4-05	7430	4142.46	12910.90	21	2	3316.57	11935.55	825.89	975.36	6	4.164
c-4-06	7300	4062.18	12565.57	18	2	3137.55	11534.38	924.64	1031.19	8	4.265
c-4-07	7365	4051.43	13337.93	20	1	3391.97	12436.15	659.45	901.78	6	4.014
c-4-08	7596	4092.15	11943.93	16	4	2885.88	10464.29	1206.27	1479.64	10	5.996
c-4-09	7433	3983.36	12835.82	19	2	3092.05	11776.49	891.31	1059.32	7	14.145
c-4-10	7237	4191.79	12403.54	19	2	3259.36	11321.67	932.43	1081.86	7	3.417
c-6-01	5806	4221.84	10267.08	23	1	3554.46	9493.57	667.38	773.51	4	7.387
c-6-02	5566	3988.72	9789.90	21	1	3320.81	9010.49	667.91	779.41	5	14.416

Table 5 continued

Instance[a]	Cost	Dist.	Time	No. SV	No LV	Dist SV	Time SV	Dist LV	Time LV	Reload	CPU[b]
c-6-03	5902	4063.85	10077.61	22	2	3428.35	9292.25	635.50	785.36	4	13.969
c-6-04	5462	3819.60	9144.25	19	2	3110.24	8328.99	709.35	815.27	7	13.010
c-6-05	5813	4191.93	10343.74	23	1	3562.71	9645.57	629.22	698.17	3	3.719
c-6-06	5711	3917.83	9662.04	21	2	3255.71	8837.89	662.12	824.15	6	5.250
c-6-07	5686	3909.46	9637.49	21	2	3258.53	8846.40	650.93	791.09	6	4.282
c-6-08	5898	4254.43	10408.74	23	1	3548.44	9608.50	705.99	800.24	4	13.768
c-6-09	5874	4128.36	9909.54	21	2	3382.48	9044.94	745.88	864.60	5	13.425
c-6-10	5721	4005.19	9737.64	21	2	3363.60	8994.59	641.59	743.05	5	7.368
r-2-01	6970	3369.99	12597.76	18	1	2765.65	11704.69	604.33	893.07	5	4.161
r-2-02	6878	3198.19	12460.41	17	1	2628.33	11603.93	569.86	856.48	6	4.109
r-2-03	6550	3264.93	11253.50	15	2	2548.44	10283.63	716.49	969.88	8	4.319
r-2-04	7027	3375.43	12848.78	20	1	2886.90	12083.30	488.53	765.48	3	3.666
r-2-05	6672	3286.40	12068.68	17	1	2712.26	11256.83	574.13	811.85	6	4.142
r-2-06	6643	3306.17	12034.93	20	1	2775.50	11237.67	530.68	797.26	3	4.018
r-2-07	6749	3177.39	12253.30	17	1	2622.63	11475.81	554.75	777.49	6	3.476
r-2-08	6798	3291.49	12371.43	20	1	2771.56	11608.55	519.92	762.88	3	3.139
r-2-09	5940	3045.85	10618.08	16	1	2503.90	9779.29	541.95	838.79	7	4.280
r-2-10	7504	3336.14	13336.67	18	2	2792.83	12499.17	543.30	837.50	6	3.914
r-4-01	6223	3144.37	10293.58	16	3	2499.69	9420.14	644.68	873.43	7	6.744
r-4-02	6028	3349.07	10904.40	17	1	2816.59	10284.10	532.48	620.29	6	4.240
r-4-03	5903	3289.00	10499.59	17	1	2663.00	9698.10	626.00	801.49	7	4.166
r-4-04	5465	3035.72	9784.28	14	1	2523.72	9121.14	512.00	663.14	8	3.982
r-4-05	5754	3093.91	10321.92	17	1	2617.45	9553.60	476.46	768.32	6	3.467

Table 5 continued

Instance[a]	Cost	Dist.	Time	No. SV	No LV	Dist SV	Time SV	Dist LV	Time LV	Reload	CPU[b]
r-4-06	5764	3172.64	10395.50	18	1	2698.87	9739.45	473.77	656.06	4	4.099
r-4-07	5986	3224.50	10866.24	19	1	2821.50	10169.90	403.00	696.33	4	4.081
r-4-08	5783	3157.47	9322.57	14	3	2442.77	8350.24	714.70	972.33	9	4.217
r-4-09	5701	3101.37	9790.94	16	2	2562.04	9050.28	539.32	740.66	7	4.016
r-4-10	5346	3200.64	9529.86	18	1	2712.37	8835.10	488.27	694.76	5	4.179
r-6-01	4428	3068.84	7817.00	19	1	2667.93	7253.28	400.91	563.72	4	13.632
r-6-02	4324	2957.61	7616.79	18	1	2580.82	7027.79	376.79	589.01	6	22.286
r-6-03	4254	2944.27	7486.13	17	1	2545.14	6937.93	399.14	548.20	7	5.258
r-6-04	4275	3024.22	7595.55	20	1	2672.20	7140.48	352.02	455.07	3	4.047
r-6-05	4189	2957.58	7401.65	20	1	2590.64	6922.98	366.94	478.67	3	6.850
r-6-06	4274	2999.90	7544.39	18	1	2591.03	7054.62	408.87	489.77	6	23.550
r-6-07	4337	3116.94	7650.62	19	1	2680.69	7168.10	436.26	482.53	4	16.078
r-6-08	4247	2986.85	7400.42	18	1	2507.73	6829.77	479.12	570.65	5	13.592
r-6-09	4263	2868.99	7083.54	17	2	2439.94	6534.80	429.05	548.75	6	17.008
r-6-10	4286	2988.18	7564.95	20	1	2596.97	7058.87	391.20	506.09	3	10.788
rc-2-01	8717	4570.47	15268.74	22	2	3705.53	13923.53	864.94	1345.21	6	4.461
rc-2-02	8958	4594.31	15634.21	21	2	3589.66	14246.50	1004.65	1387.71	7	4.263
rc-2-03	9411	4545.42	16758.91	22	2	3647.79	15667.26	897.63	1091.65	6	4.058
rc-2-04	8319	4622.86	14413.44	23	2	3632.47	13134.25	990.39	1279.19	5	3.789
rc-2-05	9032	4773.75	15790.00	22	2	3776.01	14413.06	997.74	1376.94	7	3.537
rc-2-06	9294	4602.96	15820.38	20	3	3518.02	14363.66	1084.94	1456.72	7	3.890
rc-2-07	8135	4142.41	13324.14	19	4	3193.71	12112.82	948.70	1211.32	9	4.444
rc-2-08	8563	4653.11	14910.30	20	2	3614.08	13693.28	1039.04	1217.02	7	3.885

Table 5 continued

Instance[a]	Cost	Dist.	Time	No. SV	No LV	Dist SV	Time SV	Dist LV	Time LV	Reload	CPU[b]
rc-2-09	8516	4548.09	14909.07	23	2	3616.82	13752.85	931.27	1156.21	5	4.135
rc-2-10	8820	4553.88	15375.67	20	2	3481.21	14109.14	1072.67	1266.53	8	3.828
rc-4-01	7670	4469.35	12894.43	23	3	3573.42	11842.01	895.93	1052.42	5	3.478
rc-4-02	7459	4343.97	12941.11	21	2	3478.81	11968.42	865.16	972.69	7	3.904
rc-4-03	7716	4358.50	12501.26	17	4	3386.11	11338.46	972.39	1162.80	9	7.812
rc-4-04	7609	4364.42	13208.06	22	2	3485.99	12192.17	878.44	1015.89	6	4.150
rc-4-05	7518	4429.19	12946.26	20	2	3546.78	11762.01	882.41	1184.25	8	4.451
rc-4-06	7670	4269.65	12898.60	22	3	3395.76	11817.13	873.89	1081.47	6	3.703
rc-4-07	7882	4471.73	13283.14	20	3	3565.04	12150.89	906.69	1132.25	8	4.042
rc-4-08	7429	4360.06	12840.56	20	2	3475.57	11805.44	884.49	1035.12	8	4.670
rc-4-09	8285	4797.49	14598.90	25	2	3980.70	13600.73	816.78	998.17	3	3.663
rc-4-10	7555	4501.98	13154.27	22	2	3685.55	12161.00	816.43	993.26	6	3.952
rc-6-01	6283	4386.06	10601.51	21	2	3542.73	9621.57	843.33	979.94	7	9.819
rc-6-02	5976	4297.21	10713.21	23	1	3740.07	9998.11	557.15	715.11	5	12.285
rc-6-03	5959	4294.48	10617.73	24	1	3654.57	9907.03	639.90	710.70	4	3.835
rc-6-04	5980	4296.78	10676.84	24	1	3678.33	9969.07	618.46	707.77	4	4.176
rc-6-05	6176	4343.54	10654.31	24	2	3715.69	9928.21	627.85	726.10	4	13.600
rc-6-06	6042	4382.26	10870.75	25	1	3827.09	10220.56	555.17	650.19	3	6.829
rc-6-07	6056	4033.76	9929.89	22	3	3348.14	9099.68	685.61	830.21	7	6.248
rc-6-08	5749	4165.38	10355.11	24	1	3657.54	9772.44	507.84	582.67	4	6.826
rc-6-09	6208	4330.43	10593.39	22	2	3628.33	9705.23	702.11	888.15	6	9.120
rc-6-10	5801	4122.28	10305.14	23	1	3516.89	9544.67	605.40	760.47	5	4.383

[a] Instance description: c-2-01: Instance-Set: C, Time Window: 2 h and Instance 01.

[b] CPU time in seconds

Table 6 Results VRPTW

Instance[a]	Cost	Dist.	Time	No. SV	No LV	Dist SV	Time SV	Dist LV	Time LV	Reload	CPU[b]
c-2-01	9944	5311.31	19740.34	37	0	5311.31	19740.34	0.00	0.00	0	3.044
c-2-02	10205	5333.17	20261.21	37	0	5333.17	20261.21	0.00	0.00	0	3.089
c-2-03	9659	5334.96	19170.00	37	0	5334.96	19170.00	0.00	0.00	0	2.932
c-2-04	9968	5335.94	19787.39	37	0	5335.94	19787.39	0.00	0.00	0	3.027
c-2-05	9929	5326.69	19710.96	37	0	5326.69	19710.96	0.00	0.00	0	3.105
c-2-06	10897	5343.22	21646.93	37	0	5343.22	21646.93	0.00	0.00	0	2.992
c-2-07	10336	5349.11	20524.51	37	0	5349.11	20524.51	0.00	0.00	0	3.068
c-2-08	10123	5353.15	20097.56	37	0	5353.15	20097.56	0.00	0.00	0	3.063
c-2-09	10123	5360.12	20098.36	37	0	5360.12	20098.36	0.00	0.00	0	3.013
c-2-10	10315	5338.34	20482.67	37	0	5338.34	20482.67	0.00	0.00	0	3.064
c-4-01	8334	5337.57	16520.90	37	0	5337.57	16520.90	0.00	0.00	0	3.084
c-4-02	8554	5338.00	16959.72	37	0	5338.00	16959.72	0.00	0.00	0	2.906
c-4-03	9634	5333.80	19120.06	37	0	5333.80	19120.06	0.00	0.00	0	2.986
c-4-04	8642	5345.22	17135.70	37	0	5345.22	17135.70	0.00	0.00	0	3.006
c-4-05	9004	5325.97	17860.43	37	0	5325.97	17860.43	0.00	0.00	0	3.139
c-4-06	8617	5340.49	17085.51	37	0	5340.49	17085.51	0.00	0.00	0	3.055
c-4-07	8670	5345.96	17191.03	37	0	5345.96	17191.03	0.00	0.00	0	3.007
c-4-08	8520	5333.70	16891.90	37	0	5333.70	16891.90	0.00	0.00	0	2.987
c-4-09	8515	5346.86	16882.90	37	0	5346.86	16882.90	0.00	0.00	0	2.963
c-4-10	8872	5332.80	17595.58	37	0	5332.80	17595.58	0.00	0.00	0	2.971
c-6-01	7191	5337.55	14233.46	37	0	5337.55	14233.46	0.00	0.00	0	2.706
c-6-02	7222	5293.06	14295.82	37	0	5293.06	14295.82	0.00	0.00	0	3.003

Table 6 continued

Instance[a]	Cost	Dist.	Time	No. SV	No LV	Dist SV	Time SV	Dist LV	Time LV	Reload	CPU[b]
c-6-03	7255	5317.77	14361.73	37	0	5317.77	14361.73	0.00	0.00	0	2.945
c-6-04	7254	5317.18	14360.14	37	0	5317.18	14360.14	0.00	0.00	0	2.948
c-6-05	7252	5315.51	14355.70	37	0	5315.51	14355.70	0.00	0.00	0	3.004
c-6-06	7259	5320.79	14369.76	37	0	5320.79	14369.76	0.00	0.00	0	3.132
c-6-07	7228	5297.71	14308.23	37	0	5297.71	14308.23	0.00	0.00	0	3.026
c-6-08	7245	5310.13	14341.35	37	0	5310.13	14341.35	0.00	0.00	0	2.901
c-6-09	7254	5316.92	14359.44	37	0	5316.92	14359.44	0.00	0.00	0	2.893
c-6-10	7267	5326.50	14385.00	37	0	5326.50	14385.00	0.00	0.00	0	2.928
r-2-01	7987	3736.85	15854.44	30	0	3736.85	15854.44	0.00	0.00	0	2.990
r-2-02	8533	3752.10	16945.35	30	0	3752.10	16945.35	0.00	0.00	0	3.084
r-2-03	7577	3760.02	15034.02	30	0	3760.02	15034.02	0.00	0.00	0	3.071
r-2-04	7994	3776.98	15868.12	30	0	3776.98	15868.12	0.00	0.00	0	3.036
r-2-05	7950	3775.57	15779.59	30	0	3775.57	15779.59	0.00	0.00	0	2.975
r-2-06	7846	3754.01	15572.89	30	0	3754.01	15572.89	0.00	0.00	0	3.081
r-2-07	7790	3770.92	15459.41	30	0	3770.92	15459.41	0.00	0.00	0	3.000
r-2-08	7594	3755.24	15068.04	30	0	3755.24	15068.04	0.00	0.00	0	2.891
r-2-09	7714	3780.25	15308.89	30	0	3780.25	15308.89	0.00	0.00	0	3.125
r-2-10	8662	3750.31	17204.11	30	0	3750.31	17204.11	0.00	0.00	0	3.015
r-4-01	6948	3737.75	13776.59	30	0	3737.75	13776.59	0.00	0.00	0	3.032
r-4-02	6350	3749.39	12580.85	30	0	3749.39	12580.85	0.00	0.00	0	2.983
r-4-03	6666	3702.75	13212.91	30	0	3702.75	13212.91	0.00	0.00	0	2.898
r-4-04	6356	3736.95	12592.10	30	0	3736.95	12592.10	0.00	0.00	0	2.920

Table 6 continued

Instance[a]	Cost	Dist.	Time	No. SV	No LV	Dist SV	Time SV	Dist LV	Time LV	Reload	CPU[b]
r-4-05	6445	3700.53	12769.90	30	0	3700.53	12769.90	0.00	0.00	0	3.018
r-4-06	6551	3738.62	12981.30	30	0	3738.62	12981.30	0.00	0.00	0	2.962
r-4-07	6645	3681.57	13170.75	30	0	3681.57	13170.75	0.00	0.00	0	3.047
r-4-08	6277	3714.26	12433.35	30	0	3714.26	12433.35	0.00	0.00	0	3.070
r-4-09	6614	3719.92	13108.72	30	0	3719.92	13108.72	0.00	0.00	0	2.990
r-4-10	6765	3745.60	13409.71	30	0	3745.60	13409.71	0.00	0.00	0	2.948
r-6-01	5049	3682.70	9977.55	30	0	3682.70	9977.55	0.00	0.00	0	2.896
r-6-02	5015	3657.11	9909.29	30	0	3657.11	9909.29	0.00	0.00	0	3.093
r-6-03	5015	3657.03	9909.08	30	0	3657.03	9909.08	0.00	0.00	0	2.986
r-6-04	5031	3669.10	9941.26	30	0	3669.10	9941.26	0.00	0.00	0	2.919
r-6-05	5049	3683.09	9978.58	30	0	3683.09	9978.58	0.00	0.00	0	3.053
r-6-06	5041	3677.16	9962.77	30	0	3677.16	9962.77	0.00	0.00	0	2.989
r-6-07	5036	3673.23	9952.28	30	0	3673.23	9952.28	0.00	0.00	0	3.131
r-6-08	5049	3682.79	9977.78	30	0	3682.79	9977.78	0.00	0.00	0	2.902
r-6-09	5044	3679.04	9967.77	30	0	3679.04	9967.77	0.00	0.00	0	3.097
r-6-10	5060	3691.06	9999.81	30	0	3691.06	9999.81	0.00	0.00	0	3.078
rc-2-01	9114	5161.02	18088.73	35	0	5161.02	18088.73	0.00	0.00	0	2.998
rc-2-02	9924	5166.15	19708.38	35	0	5166.15	19708.38	0.00	0.00	0	3.054
rc-2-03	9629	5104.30	19118.18	35	0	5104.30	19118.18	0.00	0.00	0	2.965
rc-2-04	9532	5164.42	18923.70	35	0	5164.42	18923.70	0.00	0.00	0	3.087
rc-2-05	9952	5182.90	19763.75	35	0	5182.90	19763.75	0.00	0.00	0	2.999
rc-2-06	9821	5118.03	19502.93	35	0	5118.03	19502.93	0.00	0.00	0	3.000
rc-2-07	9098	5150.93	18056.98	35	0	5150.93	18056.98	0.00	0.00	0	2.945
rc-2-08	9370	5135.67	18600.04	35	0	5135.67	18600.04	0.00	0.00	0	2.973

Table 6 continued

Instance[a]	Cost	Dist.	Time	No. SV	No LV	Dist SV	Time SV	Dist LV	Time LV	Reload	CPU[b]
rc-2-09	9039	5142.69	17937.72	35	0	5142.69	17937.72	0.00	0.00	0	2.969
rc-2-10	9607	5232.69	19073.86	35	0	5232.69	19073.86	0.00	0.00	0	3.025
rc-4-01	8305	5118.49	16470.76	35	0	5118.49	16470.76	0.00	0.00	0	3.056
rc-4-02	8262	5106.58	16383.31	35	0	5106.58	16383.31	0.00	0.00	0	2.978
rc-4-03	8216	5109.74	16292.99	35	0	5109.74	16292.99	0.00	0.00	0	2.980
rc-4-04	8400	5112.16	16659.29	35	0	5112.16	16659.29	0.00	0.00	0	3.122
rc-4-05	8307	4978.34	16474.24	35	0	4978.34	16474.24	0.00	0.00	0	3.013
rc-4-06	8082	5059.03	16024.68	35	0	5059.03	16024.68	0.00	0.00	0	2.925
rc-4-07	8410	5184.35	16679.14	35	0	5184.35	16679.14	0.00	0.00	0	2.947
rc-4-08	8455	5153.26	16770.57	35	0	5153.26	16770.57	0.00	0.00	0	2.949
rc-4-09	8402	5127.14	16664.52	35	0	5127.14	16664.52	0.00	0.00	0	2.946
rc-4-10	8041	5128.69	15941.59	35	0	5128.69	15941.59	0.00	0.00	0	3.048
rc-6-01	6923	5073.56	13706.50	35	0	5073.56	13706.50	0.00	0.00	0	3.046
rc-6-02	6857	5024.23	13574.95	35	0	5024.23	13574.95	0.00	0.00	0	2.910
rc-6-03	6863	5028.52	13586.37	35	0	5028.52	13586.37	0.00	0.00	0	2.967
rc-6-04	6855	5022.59	13570.57	35	0	5022.59	13570.57	0.00	0.00	0	3.006
rc-6-05	6818	4994.77	13496.40	35	0	4994.77	13496.40	0.00	0.00	0	2.999
rc-6-06	6923	5073.44	13706.16	35	0	5073.44	13706.16	0.00	0.00	0	3.062
rc-6-07	6864	5028.81	13587.15	35	0	5028.81	13587.15	0.00	0.00	0	2.950
rc-6-08	6890	5048.82	13640.52	35	0	5048.82	13640.52	0.00	0.00	0	2.995
rc-6-09	6869	5032.93	13598.14	35	0	5032.93	13598.14	0.00	0.00	0	2.967
rc-6-10	6865	5030.12	13590.64	35	0	5030.12	13590.64	0.00	0.00	0	3.138

[a] Instance description: c-2-01: Instance-Set: C, Time Window: 2 h and Instance 01.

[b] CPU time in seconds

Table 7 Results HFVRPTW

Instance^{a}	Cost	Dist.	Time	No. SV	No LV	Dist SV	Time SV	Dist LV	Time LV	Reload	CPU^{b}
c-2-01	8945	4621.62	16491.45	26	1	3953.50	15639.22	668.13	852.22	0	5.998
c-2-02	8671	4662.69	15929.26	26	1	3963.58	15089.84	699.11	839.43	0	6.008
c-2-03	8883	4681.46	15797.92	26	2	3952.66	14688.66	728.81	1109.26	0	6.005
c-2-04	9048	4610.38	16123.50	26	2	3960.83	14894.54	649.55	1228.96	0	5.998
c-2-05	9278	4646.46	16530.88	26	2	3957.84	15253.71	688.62	1277.17	0	6.006
c-2-06	9137	4656.92	16878.72	26	1	3971.60	16055.95	685.32	822.77	0	5.997
c-2-07	9265	4604.58	16594.07	26	2	3962.99	15428.35	641.59	1165.72	0	6.005
c-2-08	8892	4626.96	16390.24	26	1	3945.81	15563.73	681.14	826.51	0	5.999
c-2-09	9371	4644.12	16765.27	26	2	3976.60	15553.80	667.52	1211.47	0	6.009
c-2-10	9350	4631.48	16599.72	26	2	3957.21	15150.12	674.28	1449.60	0	6.015
c-4-01	7601	4519.59	13899.34	26	1	3953.35	13094.61	566.24	804.74	0	5.992
c-4-02	7637	4644.03	13893.65	26	1	3976.16	13074.37	667.88	819.29	0	6.004
c-4-03	8197	4527.48	15121.38	26	1	3953.28	14388.46	574.20	732.92	0	6.008
c-4-04	7488	4489.78	13709.69	26	1	3947.80	12942.77	541.99	766.91	0	5.994
c-4-05	8073	4560.16	14789.93	26	1	3937.07	13957.66	623.09	832.27	0	5.995
c-4-06	7505	4538.46	13725.70	26	1	3941.54	13000.15	596.92	725.55	0	5.999
c-4-07	7207	4572.09	13007.84	26	1	3924.89	12108.71	647.19	899.13	0	5.990
c-4-08	7281	4627.28	13194.76	26	1	3956.30	12405.97	670.98	788.79	0	6.008
c-4-09	7729	4509.85	14179.37	26	1	3942.91	13424.70	566.94	754.66	0	6.008
c-4-10	7324	4546.98	13332.20	26	1	3946.98	12547.83	599.99	784.37	0	5.996
c-6-01	6087	4466.96	11047.18	26	1	3948.13	10528.35	518.84	518.84	0	5.258
c-6-02	6162	4445.25	11174.33	26	1	3928.85	10606.93	516.40	567.40	0	5.993

Table 7 continued

Instance[a]	Cost	Dist.	Time	No. SV	No LV	Dist SV	Time SV	Dist LV	Time LV	Reload	CPU[b]
c-6-03	6112	4399.27	11134.51	26	1	3932.55	10616.79	466.72	517.72	0	5.991
c-6-04	6131	4418.75	11139.25	26	1	3923.70	10593.21	495.05	546.05	0	6.003
c-6-05	6165	4449.09	11174.48	26	1	3926.63	10601.03	522.45	573.45	0	6.019
c-6-06	6131	4416.23	11155.50	26	1	3934.97	10623.24	481.26	532.26	0	6.006
c-6-07	6121	4408.50	11133.72	26	1	3926.53	10600.75	481.97	532.97	0	6.013
c-6-08	6182	4465.34	11181.44	26	1	3921.06	10586.16	544.28	595.28	0	5.989
c-6-09	6188	4469.26	11198.46	26	1	3928.92	10607.12	540.33	591.33	0	6.005
c-6-10	6209	4486.80	11227.91	26	1	3936.06	10626.17	550.73	601.73	0	6.003
r-2-01	6970	3421.82	12659.22	23	1	2893.31	11821.74	528.52	837.48	0	6.002
r-2-02	7101	3445.95	12881.88	23	1	2878.52	12018.42	567.43	863.46	0	5.967
r-2-03	6483	3397.05	11753.89	23	1	2917.39	10986.27	479.66	767.62	0	5.984
r-2-04	7379	3429.68	13535.22	23	1	2911.37	12799.48	518.31	735.74	0	6.007
r-2-05	6698	3459.56	12092.39	23	1	2912.81	11232.81	546.75	859.57	0	5.997
r-2-06	6609	3420.06	11978.45	23	1	2922.12	11181.29	497.94	797.16	0	5.982
r-2-07	6819	3417.09	12387.21	23	1	2911.02	11577.69	506.07	809.52	0	6.028
r-2-08	6745	3414.39	12237.35	23	1	2889.82	11448.73	524.57	788.62	0	5.997
r-2-09	6552	3413.79	11818.29	23	1	2888.76	10965.03	525.03	853.27	0	6.014
r-2-10	7534	3397.98	13828.56	23	1	2888.68	13046.62	509.31	781.94	0	5.990
r-4-01	5889	3297.69	10664.54	23	1	2892.64	9988.78	405.05	675.76	0	6.028
r-4-02	5739	3306.28	10380.27	23	1	2885.17	9760.07	421.11	620.20	0	5.976
r-4-03	5839	3312.93	10514.35	23	1	2865.56	9796.91	447.37	717.44	0	5.982
r-4-04	5635	3281.73	10186.94	23	1	2893.24	9549.50	388.49	637.45	0	6.013
r-4-05	5990	3297.23	10828.57	23	1	2873.97	10100.80	423.26	727.77	0	6.010

Table 7 continued

Instance[a]	Cost	Dist.	Time	No. SV	No LV	Dist SV	Time SV	Dist LV	Time LV	Reload	CPU[b]
r-4-06	5849	3317.16	10548.00	23	1	2864.48	9865.21	452.68	682.79	0	6.023
r-4-07	5926	3234.21	10769.47	23	1	2881.36	10083.54	352.85	685.93	0	6.012
r-4-08	5476	3335.99	9823.18	23	1	2883.27	9183.69	452.72	639.49	0	5.995
r-4-09	5929	3325.34	10756.95	23	1	2898.54	10137.03	426.80	619.91	0	5.981
r-4-10	5715	3341.41	10277.99	23	1	2890.80	9588.99	450.61	689.00	0	5.954
r-6-01	4546	3222.08	8148.25	23	1	2861.50	7749.67	360.58	398.58	0	5.984
r-6-02	4520	3203.83	8090.88	23	1	2838.03	7687.07	365.81	403.81	0	5.998
r-6-03	4556	3235.79	8127.68	23	1	2840.93	7694.82	394.86	432.86	0	6.014
r-6-04	4519	3201.07	8097.88	23	1	2843.88	7702.69	357.19	395.19	0	6.018
r-6-05	4535	3215.28	8113.62	23	1	2844.80	7705.13	370.48	408.48	0	5.993
r-6-06	4586	3256.35	8200.72	23	1	2872.43	7778.80	383.92	421.92	0	6.006
r-6-07	4549	3227.05	8134.63	23	1	2850.34	7719.92	376.71	414.71	0	5.978
r-6-08	4558	3233.61	8152.75	23	1	2857.28	7738.43	376.33	414.33	0	6.023
r-6-09	4552	3229.52	8140.40	23	1	2852.33	7725.20	377.20	415.20	0	5.977
r-6-10	4493	3179.86	8057.88	23	1	2832.61	7672.63	347.25	385.25	0	6.028
rc-2-01	8604	4909.77	15681.79	28	1	4092.98	14797.34	816.79	884.44	0	5.988
rc-2-02	8664	4729.36	15908.52	28	1	4078.77	15006.28	650.60	902.24	0	5.969
rc-2-03	8637	4833.71	15825.50	28	1	4096.17	14985.76	737.54	839.74	0	5.991
rc-2-04	9009	4735.60	15962.94	28	2	4055.19	14630.20	680.41	1332.74	0	6.043
rc-2-05	9452	4720.24	16816.92	28	2	4090.43	15348.55	629.81	1468.37	0	6.005
rc-2-06	9658	4815.06	17154.45	28	2	4054.84	15718.49	760.22	1435.96	0	6.017
rc-2-07	8592	4762.89	15247.89	28	2	4082.38	14152.73	680.51	1095.16	0	5.988
rc-2-08	8561	4804.10	15669.65	28	1	4089.49	14789.08	714.61	880.57	0	5.987
rc-2-09	8589	4825.96	15201.43	28	2	4110.32	14072.63	715.63	1128.80	0	6.002

Table 7 continued

Instance[a]	Cost	Dist.	Time	No. SV	No LV	Dist SV	Time SV	Dist LV	Time LV	Reload	CPU[b]
rc-2-10	9364	4830.42	16724.73	28	2	4110.64	15547.97	719.77	1176.75	0	5.988
rc-4-01	7492	4756.15	13575.80	28	1	4074.10	12739.45	682.05	836.35	0	6.024
rc-4-02	7516	4666.94	13752.14	28	1	4062.98	13062.46	603.96	689.68	0	5.992
rc-4-03	7025	4682.72	12682.75	28	1	4064.17	11839.10	618.56	843.65	0	5.978
rc-4-04	7487	4766.94	13560.23	28	1	4074.56	12725.77	692.38	834.45	0	6.026
rc-4-05	7755	4725.67	14119.78	28	1	4070.40	13280.20	655.27	839.59	0	5.985
rc-4-06	7212	4696.83	13085.70	28	1	4066.37	12314.76	630.46	770.94	0	6.002
rc-4-07	7756	4730.43	14116.83	28	1	4060.42	13289.99	670.01	826.84	0	5.956
rc-4-08	7362	4691.49	13390.33	28	1	4071.22	12617.13	620.27	773.20	0	6.010
rc-4-09	8074	4758.93	14730.12	28	1	4067.07	13888.30	691.87	841.82	0	6.004
rc-4-10	7266	4712.68	13195.41	28	1	4089.84	12417.71	622.84	777.70	0	6.007
rc-6-01	6323	4567.43	11494.02	28	1	4049.75	10939.34	517.67	554.67	0	5.962
rc-6-02	6331	4574.19	11503.04	28	1	4051.11	10942.97	523.07	560.07	0	6.001
rc-6-03	6271	4521.82	11434.68	28	1	4041.51	10917.37	480.31	517.31	0	6.009
rc-6-04	6362	4605.79	11510.97	28	1	4036.91	10905.09	568.88	605.88	0	5.998
rc-6-05	6340	4584.19	11500.09	28	1	4043.34	10922.25	540.85	577.85	0	6.011
rc-6-06	6336	4579.66	11500.82	28	1	4046.50	10930.66	533.16	570.16	0	6.015
rc-6-07	6295	4543.70	11459.41	28	1	4043.22	10921.93	500.48	537.48	0	5.963
rc-6-08	6313	4560.81	11466.49	28	1	4037.21	10905.89	523.60	560.60	0	5.961
rc-6-09	6343	4585.82	11514.61	28	1	4051.08	10942.87	534.74	571.74	0	6.001
rc-6-10	6346	4592.76	11482.70	28	1	4027.77	10880.71	564.99	601.99	0	5.991

[a] Instance description: c-2-01: Instance-Set: C, Time Window: 2 h and Instance 01.
[b] CPU time in seconds

References

Anderluh A, Hemmelmayr VC, Nolz PC (2017) Synchronizing vans and cargo bikes in a city distribution network. Cent Eur J Oper Res 25(2):345–376

Archetti C, Speranza MG (2014) A survey on matheuristics for routing problems. EURO J Comput Optim 2(4):223–246

Ascheuer N, Fischetti M, Grötschel M (2001) Solving the asymmetric travelling salesman problem with time windows by branch-and-cut. Math Program Ser A 90(3):475–506

Baker B, Ayechew M (2003) A genetic algorithm for the vehicle routing problem. Comput Oper Res 30(5):787–800

Baldacci R, Battarra M, Vigo D (2008) Routing a heterogeneous fleet of vehicles. In: Golden B, Raghavan S, Wasil E (eds) The vehicle routing problem: latest advances and new challenges. Springer, Boston, pp 3–27

Boschetti MA, Maniezzo V, Roffilli M, Bolufé Röhler A (2009) Matheuristics: optimization, simulation and control. In: Hutchison D, Kanade T, Kittler J, Kleinberg JM, Mattern F, Mitchell JC, Naor M, Nierstrasz O, Pandu Rangan C, Steffen B, Sudan M, Terzopoulos D, Tygar D, Vardi MY, Weikum G, Blesa MJ, Blum C, Di Gaspero L, Roli A, Sampels M, Schaerf A (eds) Hybrid metaheuristics, lecture notes in computer science, vol 5818. Springer, Berlin, pp 171–177

Brandstätter C, Reimann M (2018a) Performance analysis of a metaheuristic algorithm for the line-haul feeder vehicle routing problem. J Veh Routing Algorithms 1(2):121–138

Brandstätter C, Reimann M (2018b) The line-haul feeder vehicle routing problem: mathematical model formulation and heuristic approaches. Eur J Oper Res 270(1):157–170

Bräysy O, Gendreau M (2005a) Vehicle routing problem with time windows, part i: route construction and local search algorithms. Transp Sci 39(1):104–118

Bräysy O, Gendreau M (2005b) Vehicle routing problem with time windows, part ii: metaheuristics. Transp Sci 39(1):119–139

Bredström D, Rönnqvist M (2008) Combined vehicle routing and scheduling with temporal precedence and synchronization constraints. Eur J Oper Res 191(1):19–31

Cattaruzza D, Absi N, Feillet D, Vidal T (2014) A memetic algorithm for the multi trip vehicle routing problem. Eur J Oper Res 236(3):833–848

Cattaruzza D, Absi N, Feillet D (2016) Vehicle routing problems with multiple trips. 4OR-A Q J Oper Res 14(3):223–259

Chao IM, Golden B, Wasil E (2016) A computational study of a new heuristic for the site-dependent vehicle routing problem. INFOR Inf Syst Oper Res 37(3):319–336

Cheikh M, Ratli M, Mkaouar O, Jarboui B (2015) A variable neighborhood search algorithm for the vehicle routing problem with multiple trips. Electron Notes Discrete Math 47:277–284

Chen HK (2015) Issues for the linehaul-feeder vehicle routing problem with virtual depots and time windows. J East Asia Soc Transp Stud 11:678–692

Chen HK, Wang H (2012) A two-stage algorithm for the extended linehaul-feeder vehicle routing problem with time windows. Int J Shipp Transp Logist 4(4):339–356

Chen HK, Chou HW, Hsu CY (2011a) The linehaul-feeder vehicle routing problem with virtual depots and time windows. Math Probl Eng (Article ID 759418):1–15

Chen HK, Chou HW, Hsueh CF, Ho TY (2011b) The linehaul-feeder vehicle routing problem with virtual depots. IEEE Trans Autom Sci Eng 8(4):694–704

Cordeau JF, Laporte G (2016) A tabu search algorithm for the site dependent vehicle routing problem with time windows. INFOR Inf Syst Oper Res 39(3):292–298

Cordeau JF, Gendreau M, Laporte G, Potvin JY, Semet F (2002) A guide to vehicle routing heuristics. J Oper Res Soc 53(5):512–522

Dantzig G, Fulkerson R, Hohnson S (1954) Solution of a large-scale traveling salesman problem. J Oper Res Soc Am 2(4):393–410

Dash S, Günlük O, Lodi A, Tramontani A (2012) A time bucket formulation for the traveling salesman problem with time windows. INFORMS J Comput 24(1):132–147

Deflorio F, Gonzalez-Feliu J, Perboli G, Tadei R (2012) The influence of time windows on the costs of urban freight distribution services in city logistics applications. Eur J Transp Infrastruct Res 12(3):256–274

Delorme M, Iori M, Martello S (2016) Bin packing and cutting stock problems: mathematical models and exact algorithms. Eur J Oper Res 255(1):1–20

Doerner KF, Schmid V (2010) Survey: matheuristics for rich vehicle routing problems. In: Blesa MJ (ed) Hybrid metaheuristics, lecture notes in computer science, vol 6373. Springer, Berlin, pp 206–221

Dorigo M, Blum C (2005) Ant colony optimization theory: a survey. Theor Comput Sci 344(2–3):243–278

Drexl M (2012) Synchronization in vehicle routing-a survey of vrps with multiple synchronization constraints. Transp Sci 46(3):297–316

Feo TA, Resende MGC (1995) Greedy randomized adaptive search procedures. J Glob Optim 6(2):109–133

François V, Arda Y, Crama Y, Laporte G (2016) Large neighborhood search for multi-trip vehicle routing. Eur J Oper Res 255(2):422–441

Gendreau M, Potvin JY, Bräumlaysy O, Hasle G, Løkketangen A (2008) Metaheuristics for the vehicle routing problem and its extensions: a categorized bibliography. In: Golden B, Raghavan S, Wasil E (eds) The vehicle routing problem: latest advances and new challenges. Springer, Boston, pp 143–169

Glover F (1989) Tabu search-part i. INFORMS J Comput 1(3):190–206

Glover F (1990) Tabu search-part ii. INFORMS J Comput 2(1):4–32

Golden B, Assad A, Levy L, Gheysens F (1984) The fleet size and mix vehicle routing problem. Comput Oper Res 11(1):49–66

Grangier P, Gendreau M, Lehuédé F, Rousseau LM (2016) An adaptive large neighborhood search for the two-echelon multiple-trip vehicle routing problem with satellite synchronization. Eur J Oper Res 254(1):80–91

Gurobi Optimization I (2016) Gurobi optimizer reference manual. http://www.gurobi.com. Accessed 26 June 2017

Kirkpatrick S, Gelatt C, Vecchi M (1983) Optimization by simulated annealing. Science 220(4598):671–680

Koç Ç, Bektaş T, Jabali O, Laporte G (2016) Thirty years of heterogeneous vehicle routing. Eur J Oper Res 249(1):1–21

Kritikos MN, Ioannou G (2013) The heterogeneous fleet vehicle routing problem with overloads and time windows. Int J Prod Econ 144(1):68–75

Laporte G (2009) Fifty years of vehicle routing. Transp Sci 43(4):408–416

Laporte G, Ropke S, Vidal T (2014) Chapter 4: heuristics for the vehicle routing problem. In: Toth P, Vigo D (eds) Vehicle routing, pp 87–116

Liu FH, Shen SY (1999) The fleet size and mix vehicle routing problem with time windows. J Oper Res Soc 50(7):721

Penna PHV, Subramanian A, Ochi LS (2013) An iterated local search heuristic for the heterogeneous fleet vehicle routing problem. J Heuristics 19(2):201–232

Pisinger D, Ropke S (2007) A general heuristic for vehicle routing problems. Comput Oper Res 34(8):2403–2435

Pisinger D, Ropke S (2010) Large neighborhood search. In: Gendreau M, Potvin JY (eds) Handbook of metaheuristics. Springer, Boston, pp 399–419

Prins C (2004) A simple and effective evolutionary algorithm for the vehicle routing problem. Comput Oper Res 31(12):1985–2002

Reimann M, Doerner K, Hartl RF (2004) D-ants: savings based ants divide and conquer the vehicle routing problem. Comput Oper Res 31(4):563–591

Resende MGC, Ribeiro CC (2003) Greedy randomized adaptive search procedures. In: Glover F, Kochenberger GA (eds) Handbook of metaheuristics. Springer, New York, pp 219–249

Ropke S, Pisinger D (2006a) An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transp Sci 40(4):455–472

Ropke S, Pisinger D (2006b) A unified heuristic for a large class of vehicle routing problems with backhauls. Eur J Oper Res 171(3):750–775

Russo F, Comi A (2010) A classification of city logistics measures and connected impacts. Proc—Soc Behav Sci 2(3):6355–6365 the Sixth International Conference on City Logistics

Senarclens de Grancy G (2015) Applied metaheuristics for logistical challenges in congested urban areas: three essays: Dissertation. Graz

Solomon MM (1987) Algorithms for the vehicle routing and scheduling problems with time window constraints. Oper Res 35(2):254–265

Subramanian A, Penna PHV, Uchoa E, Ochi LS (2012) A hybrid algorithm for the heterogeneous fleet vehicle routing problem. Eur J Oper Res 221(2):285–295

Thangiah SR, Osman IH, Sun T (1993) Hybrid genetic algorithm, simulated annealing and tabu search methods for vehicle routing problems with time windows. Computer Science Department, Slippery Rock University, Technical Report SRU CpSc-TR-94-27, 69

United Nations PD Department of Economic and Social Affairs (2014) World urbanization prospects: The 2014 revision, custom data acquired via website. https://esa.un.org/unpd/wup/DataQuery/. Accessed 26 June 2017

van Breedam A (1995) Improvement heuristics for the vehicle routing problem based on simulated annealing. Eur J Oper Res 86(3):480–490

Wilcoxon F (1945) Individual comparisons by ranking methods. Biom Bull 1(6):80–83