Journal on Vehicle Routing Algorithms (2018) 1:121–138
https://doi.org/10.1007/s41604-018-0009-2

REGULAR PAPER

Performance analysis of a metaheuristic algorithm for the line-haul feeder vehicle routing problem

Christian Brandstätter[1] · **Marc Reimann**[1]

Received: 20 June 2018 / Accepted: 17 July 2018 / Published online: 2 August 2018

Abstract
The line-haul feeder vehicle routing problem (LFVRP) is a rather new and therefore hardly investigated problem and can be seen as vehicle routing problem with synchronisation constraints. Synchronisation takes place if two or more vehicles (small and large vehicles) meet at the same place and at the same time and perform a transshipment of goods. Small vehicles are used to serve customers which are difficult to access (e.g. due to traffic jams, narrow streets or limited parking space). The downside is their limited capacity. To avoid the journey back to the depot, the small vehicle can meet with a large vehicle at a customer with enough space for reloading. Thus, the small vehicle uses the large vehicle as virtual depot and is able to save time and reduce costs. In Brandstätter and Reimann (Eur J Oper Res 270:157–170, 2018), we already introduced two promising solution approaches to master the LFVRP—the *Linkage*- and *Split-Approach*. Both approaches were shown to find high-quality solutions in short time. Moreover, we demonstrated the advantage of the LFVRP strategy when compared with the heterogeneous fleet vehicle routing problem. With this paper, we propose and justify several improvements to our original algorithms. These include the application of Ant Colony Optimization, different local search operators, as well as the exact solution of a sub-problem. To highlight the performance contribution of these modifications, we perform a thorough and extensive computational experiment. We evaluate each algorithmic component individually as well as their combination and statistically validate the benefits. Overall, we find that the final algorithm improves our previous approaches by almost 9% at a computational cost of a few seconds.

Keywords Vehicle routing problem · Synchronisation · ACO · Metaheuristics · Matheuristics · Multiple solutions · Local search

1 Introduction

The influx of new residents is a common phenomenon for major cities around the world. According to statistics from the United Nations [46], European cities like Stockholm (27%), Madrid (20%), Munich (20%) or London (19%) will have an increase in population of more than 20% (from 2005) until 2020. With more and more people living in the city centre, new challenges for city logistics arise. Some of these challenges are increasing demands, crowded streets with hardly any parking space and also increasing land prices for buying or renting space. Thus, a depot located far outside of town might be considered as well as the use of smaller vehicles. However, smaller vehicles do also have a limited capacity and more vehicles need to be dispatched from the depot which will result in higher costs.

Another prominent problem is the particulate matter pollution, which is addressed, especially by environmental and governmental institutions (see [22]). Their aim is to hold or better reduce the level of pollution to a minimum. Hence, some governmental institutions (e.g. in Germany—Stuttgart and Munich) plan to ban motorised vehicles on several days if a critical pollution level is reached. The ban will be relevant for vehicles that do not meet certain emissions standards (e.g. Euro 6). In other words, most city logistics companies will not be allowed to deliver goods with motorised vehicles on days with a high particulate matter level. In conclusion, city logistics companies will have to master the before mentioned challenges to maintain their competitive edge.

✉ Marc Reimann
marc.reimann@uni-graz.at

Christian Brandstätter
christian.brandstaetter@edu.uni-graz.at

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

Besides the challenges for city logistics, new opportunities for vehicle routing research arise as well. New problems will be defined and analysed so that the findings can be used by city logistics companies to remain competitive. One of these problems is the line-haul feeder vehicle routing problem (LFVRP) which can be described as follows. The LFVRP consists of a single physical depot (PD), a set of customers with different characteristics in terms of parking space and a heterogeneous fleet of vehicles. The heterogeneous fleet uses two types of vehicles which are dispatched from the PD. The two vehicle classes used are categorized in terms of size, costs and vehicle load. Specifically, the large vehicle (LV) class is represented by a truck with a trailer and the small vehicle (SV) class can be a (electric) car, motor-bike or cargo bike. Furthermore, the majority of customers (type-2) can only be delivered by the SV class, whereas the remaining customers (type-1) can be delivered by both vehicle classes. The downside of using a smaller vehicle class is the limited capacity. Consequently, the smaller vehicles would need to return to the depot more often for reloading. To avoid travelling back to the depot, the small vehicle can meet with a large vehicle at a customer with enough space (type-1) for reloading (transshipment of goods). Thus, the large vehicle is acting as a virtual depot (VD) for the small vehicle class. However, both vehicles need to be at the same place at the same time to be able to perform a transshipment of goods. In other words, both vehicle tours need to be synchronised and therefore the LFVRP can be seen as a VRP with synchronisation constraints.

The purpose of this paper is to improve our previously presented heuristic approaches (see [5]) by combining the principles of metaheuristics (particularly Ant Colony Optimization), the exact solution of a sub-problem, the systematic use of multiple solutions and local search. We will present the improvement for each step supported by numerical results and also provide a statistical analysis to validate our approaches. The remainder of this paper is organized as follows. In Sect. 2, we will conduct a literature review on the methods used to improve our heuristics. A detailed description of the improvement steps is presented in Sect. 3. In Sect. 4, we will report the computational results using an extended test instance set and conclude the paper with our final remarks.

2 Literature review

The LFVRP originated in Taiwan during the delivery of lunch boxes and was first introduced by Chen et al. [14]. They presented two simple heuristics (cost-sharing and threshold method) which were tested on 17 test instances. Chen et al. [13] added time windows and provided a two-stage heuristic by using tabu search (TS) in the second stage. The initial solution of the first stage was then improved by Chen and Wang [15]. Chen [12] exchanged the original test instances to 15 test instances derived from the well-known Solomon instance set and analysed four different issues on the LFVRP. These four issues were different solution algorithms, customer demands, VD candidates and range of time windows. However, no mathematical model formulation has been provided so far.

Brandstätter and Reimann [5] formally defined the problem and solved the problem optimally for a small number of customers. Furthermore, they presented two promising solution algorithms for larger problem sizes (named the *Linkage-Approach* and *Split-Approach*) and tested them on the original test instances provided by Chen et al. [14]. While the *Linkage-Approach* tries to link SV tours through a VD, the *Split-Approach* tries to split a giant tour into feasible subtours. In addition, the *Linkage-Approach* uses a Bin Packing Problem (BPP) strategy where the bin size is equivalent to the maximum allowed working time. Other well-known bin packing algorithms are *First Fit (FF)*, *Best Fit (BF)* and *Worst Fit (WF)*. For these algorithms also, decreasing versions exist (the abbreviations will be extended with the letter D: e.g. *First Fit Decreasing (FFD)*). Interested readers are advised to refer to Johnson [28], Yao [49] or Delorme et al. [19].

The synchronisation element distinguishes the LFVRPT from other VRP variants. Yet there are some variants with rather close similarities. One of these problems is the site-dependent vehicle routing problem (SDVRP). The attributes of the SDVRP are similar in terms of a mixed fleet of vehicles and a predefined customer–vehicle relationship. In other words, some vehicles can visit only a certain customer category. For further reading on the SDVRP, the interested reader is advised to see Chao et al. [10], Cordeau and Laporte [17] or Pisinger and Ropke [37].

Another similar VRP variant is the vehicle routing problem with Multiple Trips (MTVRP). In the classical VRP, only one single tour per vehicle is allowed. However, this does not match many real-life situations. For instance, if a vehicle has a small capacity, it can only serve a limited number of customers. Hence, the vehicle will return to the depot early and can possibly start another tour. Therefore, it is possible that a vehicle performs several tours within a certain time limit. Further information on the MTVRP can be found in Cattaruzza et al. [8,9], Cheikh et al. [11] and François et al. [23]—often the problem is also referred as Multi Trip VRP, VRP with multiple routes or VRP with multiple use of vehicles.

The heterogeneous fleet vehicle routing problem (HFVRP) is another similar VRP variant where at least two different vehicle classes are used. The vehicles can differ in several aspects like size, costs, capacity or range. An overview on the HFVRP can be found in Subramanian et al. [45], Penna et al. [36], Kritikos and Ioannou [31] Koç et al. [30] and Baldacci et al. [3]. Earlier research with a heterogeneous fleet

is named Fleet Size Mix (FSM) and can be found in Golden et al. [26] or Liu and Shen [35].

Overall, there is a vast range of heuristics to solve and improve the VRP and its variants (see [16,34]). Early heuristics focused on a two-step approach that construct an initial feasible solution in the first stage and try to improve it afterwards. In the last two decades, the focus was primarily set on *metaheuristics* and for a solid overview the interested reader is advised to see Bräysy and Gendreau [6,7].

According to Laporte [33], *metaheuristics* focus on three basic principles—local search, population search and learning mechanisms. The *local search* principle searches the solution space by moving to another (better) solution in the neighbourhood at every iteration. Prominent examples are TS (see [24,25]), simulated annealing (SA) (see [29,47]) and adaptive large neighbourhood search (for ALNS see [41,42]). Genetic algorithms (GA) are famous representatives of the population search principle (see [2,39]). Here, a population of solutions is used to create offspring by using parent solutions from the population and recombining them.

A prominent representative of a learning mechanism is ant colony optimization (ACO). The basic principle behind ACO is to imitate the behaviour of ants during their search for food. When ants are deployed from their home to search for food, they lay a pheromone trail. The pheromone value of the trail will be increased (accumulated) if another ant uses that trail or decreased (evaporated) if no ant uses that trail. When returning home, an ant will choose a trail more likely with a higher pheromone value as it indicates (a possible) shorter trail. A successful implementation of ACO can be found in Reimann et al. [40], and a solid overview is given in Dorigo and Blum [21].

The research field of *matheuristics* is a rather young field of research. The basic idea is to combine exact solution techniques with heuristics. The result is a hybrid heuristic which uses the benefits of both solution techniques. A recent conducted survey by Doerner and Schmid [20] indicates that existing hybrid heuristics mostly fit into three categories—set covering, local branching and decomposition techniques. For the improvement of our algorithm, we will use the decomposition technique and describe it in more detail in the next section. For further reading on *matheuristics*, see Boschetti et al. [4], Subramanian et al. [45] or Archetti and Speranza [1].

Large neighbourhood search heuristics are often used to improve the final solution. Specifically, the neighbourhood of a solution is searched for a superior solution by using different neighbourhood search procedures. One of these procedures is named destroy and repair. The basic concept is to destroy the solution and repair it afterwards by alternating some characteristics. In the literature, this is often referred to as the large neighbourhood search and the interested reader is advised to refer to Ropke and Pisinger [41,42] and Pisinger and Ropke [37,38] for a solid overview.

3 Algorithm improvement

To start out, let us first formally define the LFVRP. The LFVRP can be modelled through a graph consisting of a set of nodes V representing one physical depot (node 0) as well as a set of customers C—which is divided into type-1 (set A) and type-2 (set B) customers—and a set of edges E which correspond to the travel links between any two nodes i and j. With each edge, a travel cost c_{ij} and time t_{ij} are associated. The customers have a demand d_i and service time t_i^S. Further, a fleet F of vehicles k with a max. vehicle capacity Q^k, which is also divided into small F_{SV} and large vehicles F_{LV}, is considered.

All vehicle tours must start and end at the physical depot (PD), and the duration of each tour must be within the given time limit. Large vehicles are only allowed to visit customers of type-1, whereas small vehicles can visit both customer types. Visiting the customers in the LFVRP is different than in the usual VRP variants. Usually, customers may only be visited once, but for the LFVRP that is only true for customers of type-2. Type-1 customers can be visited multiple times (and by different vehicles) because they provide enough parking space, so that a large vehicle can act as a virtual depot (VD). In other words, a large vehicle can park at a customer of type-1 and meet with different small vehicles to perform multiple transshipments. A transshipment between a large and a small vehicle requires a transshipment time of t_{VD}^{TS}.

3.1 Previous work

The two LFVRP approaches we will improve are called the *Linkage-* and *Split-Approach*. For a better understanding of our different improvement steps, we will describe the two approaches briefly. For the general introduction of the LFVRP together with the formal definition and a more detailed description of both approaches, the interested reader is advised to see Brandstätter and Reimann [5].

3.1.1 Linkage-Approach

The *Linkage-Approach* consists of four different steps. In the first step, we create a VRP solution for all type-2 customers by using Solomon's I1 heuristic (see [44]: with $\alpha = 1, \lambda = 2$ and $\mu = 1$). In the next step, we try to link the existing tours through a potential VD—customer of type-1—or the PD. A linkage will be accepted if the resulting costs are less than the sum of the initial costs. That step will be repeated until no further linkage is possible. For a better understanding, we provided a small illustrative example in Fig. 1.

Let us assume that the VRP solution of Solomon's I1 heuristic is two SV tours: *SV Tour 1* and *SV Tour 2*. We now try to link the two tours through a VD. In other words, the two edges $e_{4,PD}$ and $e_{PD,5}$ will be replaced by $e_{4,VD}$

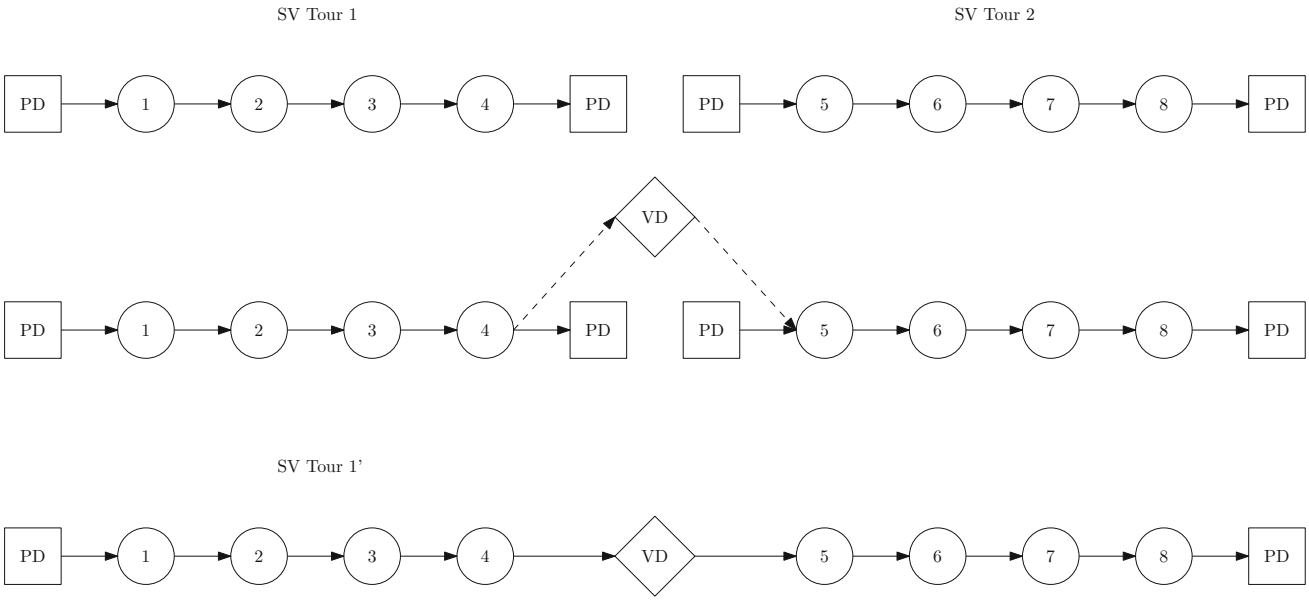


Fig. 1 First two steps of the basic *Linkage-Approach*

and $e_{VD,5}$. The two tours will be linked because the overall costs for using a VD are less than driving back to the PD ($c_{4,VD} + c_{VD,5} < c_{4,PD} + c_{PD,5}$). Consequently, we get one single SV tour (*SV Tour 1'*) with a VD in place to satisfy all the necessary constraints. In our example, we have only considered the forward version of the SV tours. Nevertheless, we can also consider the backwards version of the tours because we are not considering time windows yet. Hence, four possible tour combinations will be checked for feasibility.

In the third step, we create the LV tours. To make a transshipment possible, the small and large vehicle need to be at the same place at the same time. If tours are linked through a VD in the second step, the time and place for the transshipment is already set. Therefore, we sort all inserted VDs in their chronological order and insert them into the LV tours. In the final step, the remaining customers of type-1 will be inserted according to Solomon's I1 heuristic.

3.1.2 Split-Approach

The *Split-Approach* differs from the *Linkage-Approach* only in the first two steps. In the first step, we will solve the TSP instead of the VRP for all type-2 customers. The result is one giant tour because we neglect the capacity and time constraint for now. In the next step, we try to split the giant tour into feasible sub-tours. We start by selecting a split position in the giant tour where the time constraint (maximum allowed tour length) is satisfied. However, selecting the split position is not enough, as the capacity constraint needs to be satisfied as well. Hence, we calculate the required number of reloads and insert all necessary VD or PD visits at all feasible positions. If no feasible sub-tour solution is found, the split position will be moved to the previous customer. Now, the required number of reloads is recalculated and the insertion of the VD or PD visits is started again. That procedure will be repeated until a feasible solution is found. The sub-tour solution with minimum resulting costs will be split from the giant tour and that procedure will continue until no more split or VD insertion is necessary. To get a better understanding of the *Split-Approach*, a small illustrative example is provided in Fig. 2.

Here, the result of the first step is a giant tour with 8 customers. The split position where the time constraint is satisfied is marked between customers 4 and 5. To satisfy the capacity constraint, one reload (VD) is necessary and will be inserted between customers 1 and 2, customers 2 and 3 (remove edge $e_{2,3}$ and add edges $e_{2,VD}$ and $e_{VD,3}$) or customers 3 and 4. In our example, the VD position with the minimum cost is between customers 2 and 3. Hence, the sub-tour $PD \rightarrow 1 \rightarrow 2 \rightarrow VD \rightarrow 3 \rightarrow 4 \rightarrow PD$ is split from the giant tour. The remaining giant tour is now $PD \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow PD$ and requires only one reload to become feasible. Thus, a VD will be inserted between customers 6 and 7. As a result, the giant tour was split into two sub-tours with one inserted VD each.

3.2 Improvement approach

Our improvement approach will feature the benefits from *metaheuristics*, *matheuristics*, generating *multiple solutions* and *local search*. For a better understanding of the different strategies, we provide a comparison in Fig. 3. The four-step benchmark algorithm, which was briefly described before,

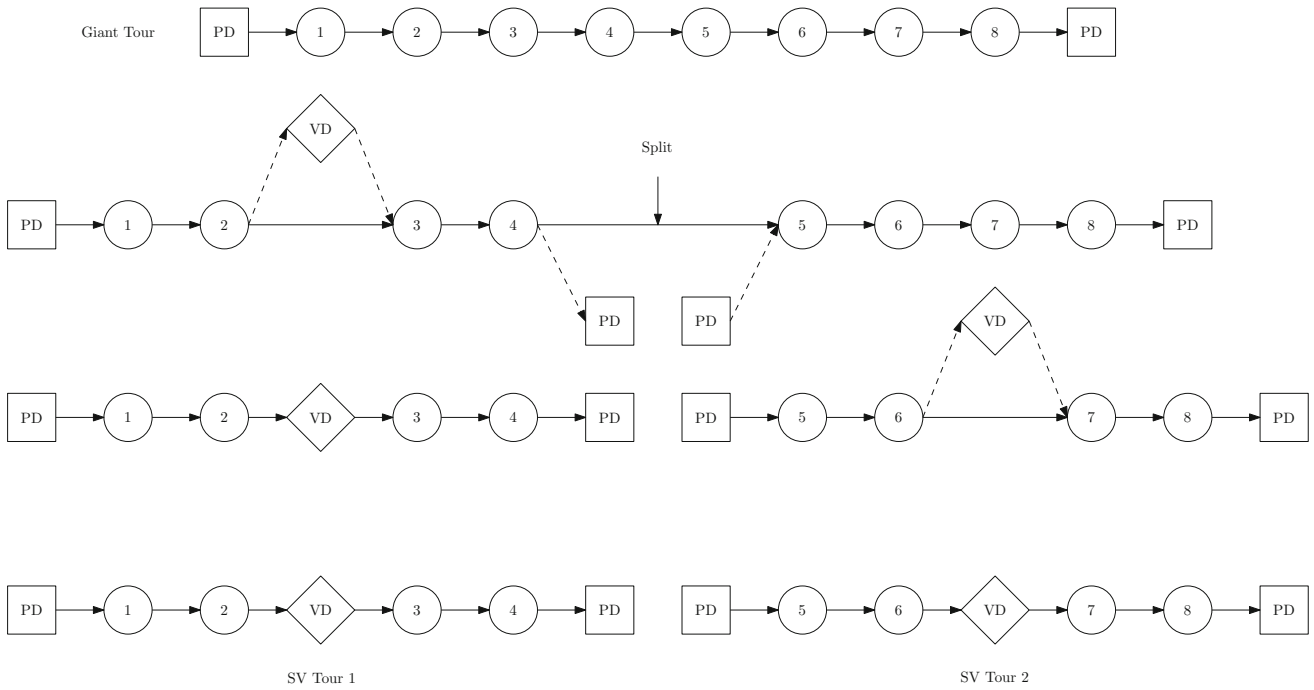


Fig. 2 First two steps of the basic *Split-Approach*

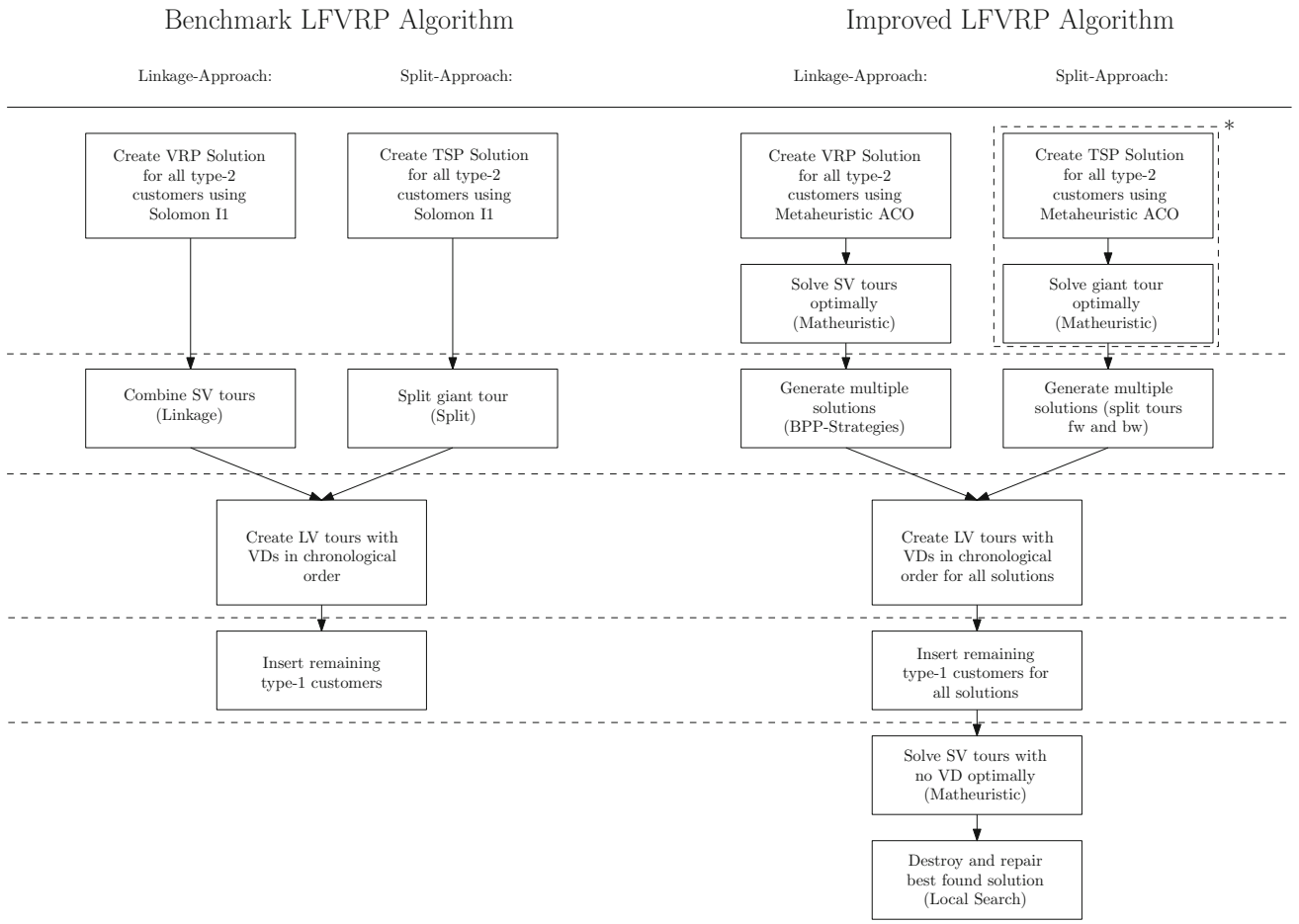


Fig. 3 Algorithm comparison

is presented on the left-hand side, whereas the improvement approach is presented on the right-hand side.

The improvement approach can be summarized as follows. We used a metaheuristic (ACO) in the first step to improve our initial solution. In the next step, we solved the SV tours (*Linkage-Approach*) and the giant tour (*Split-Approach*) optimally by using a mathematical problem solver. Afterwards, we generated multiple solutions by considering forwards and backwards versions of the tours along with the first and second best choice for a transshipment customer. Furthermore, we adapted and applied some well-known BPP strategies but only for the *Linkage-Approach*. After that, we generated the LV tours and chose the best solution (minimum cost) among all feasible solutions. In the last and final step, we applied the principle of local search to improve the best found solution. The pseudo-code for the improvement approach is presented in Algorithm 1.[1]

Algorithm 1: Improvement Approach

```
1  x = 0 For Linkage-Approach / x = 1 for Split-Approach
   (x ∈ {0, 1});
2  if x = 0 then
3      solve VRP for all type-2 customers with ACO;
4      solve all SV tours optimally;
5      apply core algorithm of Linkage-Approach: Algorithm 2;
6  else
7      solve TSP for all type-2 customers with ACO;
8      OR solve TSP for all type-2 customers optimally;
9      apply core algorithm of Split-Approach: Algorithm 3;
10 end
11 for all small vehicle tours do
12     insert all VDs into LV tours in chronological order;
13     insert all remaining type-1 customers into SV/LV tours;
14     solve all LV tours without synchronization optimally;
15     store solution in sol;
16 end
17 choose the best solution (min cost) among all solutions (sol);
18 apply local search procedure: Algorithm 4;
```

If we have a look at the improved approach in Fig. 3, we can see that the complexity of the algorithm increased significantly. Thus, we will describe each improvement strategy in more detail to better understand the interaction between them.

3.3 Metaheuristics strategy (ME)

In the first step of the benchmark algorithm, a VRP solution for the *Linkage-Approach* and a TSP solution for the *Split-Approach* for all type-2 customers were generated. Consequently, our first aim is to improve the VRP/TSP solution and for that more than half a century of scientific research is at our disposal (see [33]).

As metaheuristics are among the most promising algorithms (see [16,33]), we decided to use a metaheuristic to improve the solution quality of the first stage. We tested different metaheuristics (GRASP, TS and ACO) but with no significant difference in terms of solution quality. Due to earlier positive experience (e.g. [40,43]), we decided to use ACO for our first improvement measure. We used the ACO implementation for the vehicle routing problem with time windows (VRPTW) of Senarclens de Grancy [43] with slight adjustments. For instance, we had to disable the capacity and time limitation to gain the TSP tour for the *Split-Approach*, but for a better understanding, we will describe the ACO metaheuristic in more detail.

The basic concept of ACO is to imitate the behaviour of real ants during their search for food. The chosen path of an artificial ant is—from a VRP perspective—the series of chosen edges between the customer nodes. Thus, feasible paths (solutions) for a number of artificial ants (m) are generated by applying the principle of Solomon's I1 heuristic. More specific, all vehicle tours are constructed sequentially. A seed customer is selected and inserted between the start- and end-depot which results in one route with a single customer. After the seed customer was inserted to the route, the other customers are inserted afterwards until no further feasible insertions (in terms of tour duration and capacity) are possible. In that case, a new route is started with a new seed customer and the insertion procedure is repeated until all customers are serviced. To select a customer, the ACO metaheuristic uses a probabilistic choice. The probability P_{jkl} of inserting customer k between customers j and l is:

$$P_{jkl} = \frac{\kappa_{jkl}}{\sum_{i \in I} \kappa_i} \tag{1}$$

where (κ) represents the attractiveness for inserting a customer and I the set of possible insertions. As described before, the construction heuristic requires two types of decisions. First, the attractiveness of selecting a seed customer is calculated as:

$$\kappa_{jkl} = c_{jk}(\tau_{jk} + \tau_{kl}) \tag{2}$$

In Equation (2), j and l represent the start- and end-depot, c_{jk} the travel time/cost between the depot and the seed customer and (τ) the pheromone level of the respective edge. If a route is already initialized with a seed customer, additional customers can be added as long as feasibility is observed. Therefore, the second decision considers the insertion of a customer to an already existing route. The attractiveness for inserting customer k between customers j and l is therefore calculated as:

[1] If the matheuristic strategy is enabled within the *Split-Approach* (marked with * in Fig. 3), the metaheuristic strategy becomes obsolete because the giant tour can be solved optimally.

$$\kappa_{jkl} = \frac{1}{\overline{cost_{jkl}}} \tag{3}$$

$$cost_{jkl} = \begin{cases} \delta_{jkl} T_{jkl} & \text{if } \delta_{jkl} \leq 0 \\ \delta_{jkl} / T_{jkl} & \text{else} \end{cases} \tag{4}$$

The costs of insertion are calculated as the additional distance costs $\delta_{jkl} = c_{jk} + c_{kl} - c_{jl} - 2c_{0k}$ multiplied by the pheromone trail value $T_{jkl} = \frac{\tau_{jk}+\tau_{kl}}{2\tau_{jl}}$ if $\delta_{jkl} \leq 0$. Otherwise, the additional distance costs will be divided by the pheromone trail value (see Equation (4)). To avoid a division by 0, the costs of insertion are normalized ($\overline{cost_{jkl}}$).

After a feasible solution is found for every artificial ant, the first iteration is finished. Now the pheromone level (τ) of each edge is updated (increased if an edge is used or decreased if not used) and the next iteration starts. Prior to the first iteration, the pheromone levels are initialized with the value 1.

$$\tau_{ij} = 1, \text{for } (i, j) \in E \tag{5}$$

After each iteration, the pheromone levels are updated using the following update rule:

$$\tau_{ij} = \tau_{ij}\rho + (1 - \rho)\Delta\tau_{ij} \tag{6}$$

If an artificial ant uses an edge, the pheromone level of that edge will be increased. However, pheromone evaporates over time which results in a reduction in the pheromone level. That characteristic will be considered with the pheromone persistence factor (ρ). To update the pheromone level, it will be multiplied with the pheromone persistence factor and increased by $(1-\rho)$ if the edge e_{ij} is used ($\Delta\tau_{ij} = 1$) within the global solution. Once the pheromone level is updated the construction heuristic starts again using already the new information about the previous chosen paths. That procedure will continue until a certain stopping criterion is reached—maximum number of iterations or time limit reached. In the last step, a local search procedure is applied to improve the best found solution. For more detailed information about the design of the used metaheuristic (ACO),[2] the interested reader is advised to see Senarclens de Grancy [43].

3.4 Matheuristics strategy (MA)

After improving the starting solution with ACO, we applied a matheuristic strategy to further improve the solution. Specifically, we used the decomposition technique where a (smaller) sub-problem is extracted from the main problem. The sub-problem can then be solved optimally by using a mathematical problem solver. In the LFVRP context, the SV tours of the *Linkage-Approach* and the giant tour of the *Split-Approach* are considered as sub-problems. Therefore, for both approaches a sub-problem is one single tour with a limited no. of customers. In other words, we need to solve the Traveling Salesman Problem (TSP: see [18,32]). As the sub-problems can be considered rather small (only a few customers per tour), the TSP is solved rather quickly—far less than 1 second for the *Linkage-Approach* and less than 3 seconds (for 100 customers) for the *Split-Approach*. Thus, we also applied the matheuristic strategy to the LV tours, but only those with no synchronisation between vehicles. That is because the time and location of the VDs in the LV tours are already predefined by the SV tours. Hence, we have to accept possible detours of the LV to preserve the synchronisation between the vehicles. To solve the sub-problems we used the formulation of Dantzig et al. [18] with the sub-tour elimination constraint (SEC) $\sum_{i,j\in S, i\neq j} x_{ij} \leq |S| - 1, \quad \forall S \subset V, S \neq 0$. The TSP was then solved with the software Gurobi Optimizer.[3]

3.5 Multiple solutions strategy (MS)

In Sect. 3.1, we already presented the original designed strategies for the *Linkage-* and *Split-Approach*. However, the final solution of both approaches usually has one or more VD visits. In other words, small and large vehicle tours are synchronized and therefore interdependent. Due to this interdependence, it is rather difficult to apply an improvement step at the end of the algorithm. Therefore, we decided to apply a multiple solution strategy and choose the best solution among them.

For the *Linkage-Approach*, we will use the concept of bin packing strategies, whereas for the *Split-Approach* multiple solutions will be generated for every tour split. As we do not use time windows, we can consider all SV tours in both directions—forwards (FW) and backwards (BW). Furthermore, we will also choose the first and second best fit for a VD. To get a better understanding of both strategies, we will describe both of them in more detail.

3.5.1 Linkage-Approach

As mentioned before, we used the basic concept of the three most common bin packing strategies named *First Fit (FF)*, *Best Fit (BF)* and *Worst Fit (WF)*. For these strategies, also decreasing versions exist where all elements are first sorted in decreasing order before the bin packing strategy starts. For

[2] The configuration details for our algorithm are as follows: (i) stochastic Solomon: seed customer = random, $\alpha = 1$, $\lambda = 2$, $\mu = 1$; (ii) ACO: no. of ants = dynamic, $\rho = 0.975$, runtime limit: 3 sec./instance; (iii) local search: enabled—3 operators: shake-reduce, exchange and relocate.

[3] Version 6.5.1: for detailed information please refer to Gurobi Optimization [27].

our algorithm, the maximum tour duration is equal to the bin size in the bin packing strategy. In the decreasing version, the tours are sorted according to their tour length. How these strategies are applied to our algorithm will be described in the following paragraphs.

The FF algorithm takes the first SV tour and tries to link it with the second through a potential VD or PD. As mentioned before, not only the forwards but also the backwards version of the tours is checked. Therefore, for two tours we already have to consider four combinations. These combinations are presented in Fig. 4. In our example, we have four SV tours and the first tour cannot be linked with another tour. Therefore, the second tour will be checked for a possible linkage. A linkage with the third tour would exceed the maximum allowed tour duration and therefore we check the possible linkage with the fourth tour. The forwards and backwards versions of the second tour will be checked with the forwards and backwards versions of the fourth tour—thus four combinations. These four combinations will be checked with the best and second best VD in terms of detour, which means that finally 8 combinations are possible. If more combinations are feasible, the combination with the minimum resulting costs will be considered. The two tours will be linked together if the resulting costs are less than the separated tours together and are henceforth considered as one single SV tour. However, in the FF algorithm we do not check the other tours if a linkage of two tours is already possible. The first feasible linkage will be accepted and thus first fit. That procedure will repeat until all tours are checked and no further linkage is possible.

The BF/WF algorithm follows the same principle as the FF but with one exception. The linkage of a tour is checked with all existing tours. Therefore, if we consider our example from before, we have 16 possible tour combinations. Both tours (*SV Tour 2* and *SV Tour 4*) as starting tour with forwards and backwards consideration and also the check with best and second best VD. Again, the linked tours must be feasible together and depending on the bin packing strategy the best/worst combination will be considered.

However, we had to adjust the bin packing strategies slightly. We cannot simply add the total travel times of the tours because we have to eliminate the travel time from the last customer of the first tour to the PD and the travel time from the PD to the first customer from the second tour. In exchange for that, we have to add the travel times from and to the VD as well as the transshipment time (t^{TS}_{VD}). Finally, the combined tour duration must fit into the time limit. Let $j(k)$ denote the last (first) customer on the first (second) tour. Further let t^{SV1} and t^{SV2} denote the durations of the two tours and let t^{SV} denote the new travel time of the linked tour. The two resulting equations are presented in Eqs. 7 and 8, respectively.

$$t^{SV} = t^{SV1} - t_{j,PD} + t_{j,VD} + t^{SV2} - t_{PD,k} + t_{VD,k} + t^{TS}_{VD} \tag{7}$$

$$t^{SV} \leq \text{driver working time} \tag{8}$$

If we look again at our example in Fig. 4, the first linkage to be considered would be the forwards consideration of *SV Tour 2* and *SV Tour 4*. The new tour duration of the linked tour is calculated in Eq. 9, and if Eq. 10 is satisfied, the linkage is feasible for further consideration.

$$t^{SV} = t^{SV2} - t_{12,PD} + t_{12,VD^1} + t^{SV4} - t_{PD,19} + t_{VD^1,19} + t^{TS}_{VD^1} \tag{9}$$

$$t^{SV} \leq \text{driver working time} \tag{10}$$

To sum up, we use our original proposed strategy together with three most common bin packing strategies with random and decreasing order of the initial solution and also a forwards and backwards consideration of our tours. In addition, we also take the first and second best transshipment customer (VD) into consideration. As a result, we generate multiple solutions for the synchronisation with the LV tours. The pseudo-code for the core algorithm of the *Linkage-Approach* is presented in Algorithm 2.

Algorithm 2: Core algorithm for the *Linkage-Approach*

```
for first, second best VD or PD do
    for original and all BPP strategies do
        while SV tours can be linked through a transshipment do
            if linkage is feasible (depending on the strategy) then
                link tours with a VD/PD;
            else
                continue;
            end
        end
        save solution temporarily;
    end
end
```

3.5.2 Split-Approach

For the *Split-Approach*, we also applied the principle of a multiple solution strategy by considering the forwards and backwards versions of the giant tour and of the sub-tours, respectively. In addition, we generate solutions with the first and second best VD as transshipment customer. For a better understanding, we provide a small illustrative example in Fig. 5.

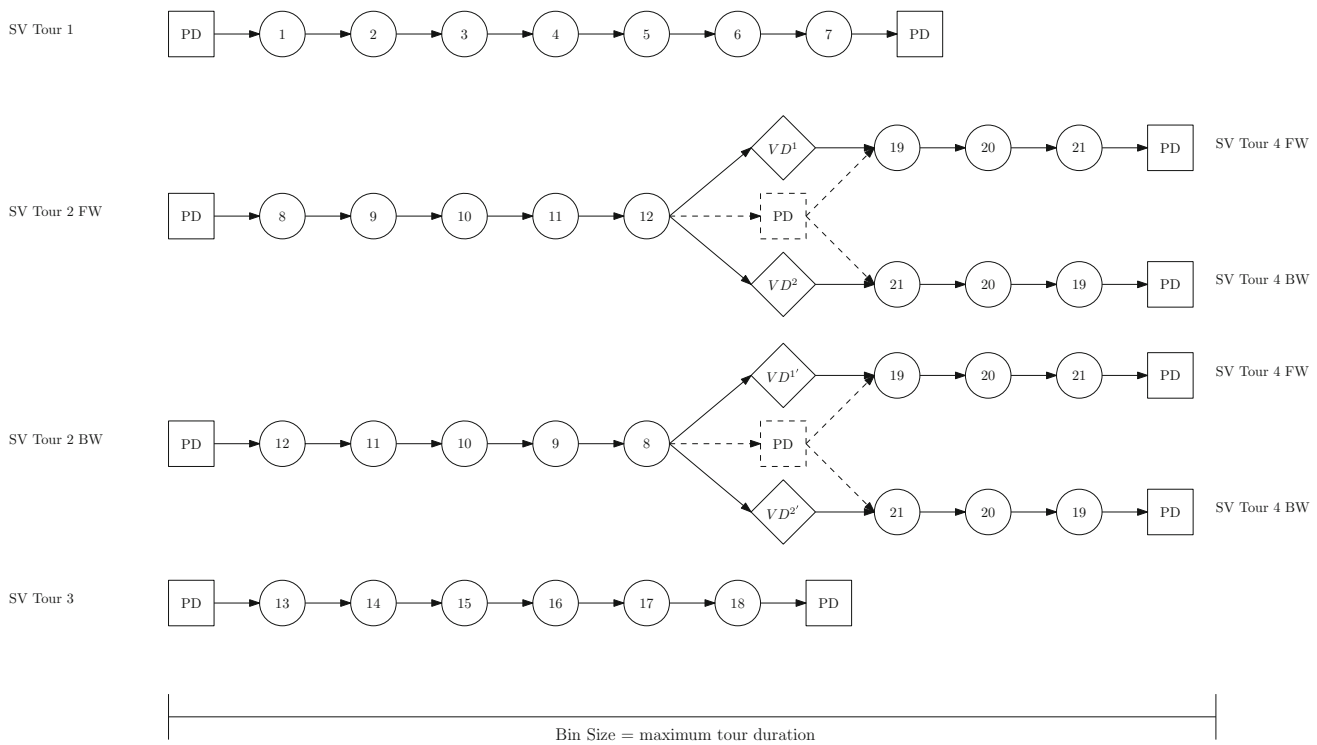


Fig. 4 *Linkage-Approach* multiple solutions

A giant tour with 14 customers will be split after the first five customers. For simplification, we make the following assumption. *Split Tour 1* has already inserted all necessary VDs to become feasible (as provided in Fig. 2). The remaining giant tour (*Giant Tour'*) is still not feasible and therefore needs to be split again. However, from this point we have to consider the forwards (FW) and backwards (BW) versions of *Split Tour 1* and also of the remaining giant tour (*Giant Tour' FW* and *Giant Tour' BW*). In other words, at this point we have four possible combinations. But now we have to split the forwards version of the remaining giant tour into *Split Tour 2* and *Split Tour 3* as well as the backwards version into *Split Tour 2'* and *Split Tour 3'*.

Figure 5 shows a giant tour with the necessary split positions. Thus, we have three SV tours for consideration. However, multiple combinations of these three tours are possible. For example, if we consider only the forwards version of the initial giant tour, we get 16 temporary solutions. These 16 solutions are as follows:

1. *Split Tour 1 FW + Split Tour 2 FW + Split Tour 3 FW*
2. *Split Tour 1 FW + Split Tour 2 FW + Split Tour 3 BW*
3. *Split Tour 1 FW + Split Tour 2 BW + Split Tour 3 FW*
4. *Split Tour 1 FW + Split Tour 2 BW + Split Tour 3 BW*
5. *Split Tour 1 FW + Split Tour 2' FW + Split Tour 3' FW*
6. *Split Tour 1 FW + Split Tour 2' FW + Split Tour 3' BW*
7. *Split Tour 1 FW + Split Tour 2' BW + Split Tour 3' FW*
8. *Split Tour 1 FW + Split Tour 2' BW + Split Tour 3' BW*
9. *Split Tour 1 BW + Split Tour 2 FW + Split Tour 3 FW*
10. *Split Tour 1 BW + Split Tour 2 FW + Split Tour 3 BW*
11. *Split Tour 1 BW + Split Tour 2 BW + Split Tour 3 FW*
12. *Split Tour 1 BW + Split Tour 2 BW + Split Tour 3 BW*
13. *Split Tour 1 BW + Split Tour 2' FW + Split Tour 3' FW*
14. *Split Tour 1 BW + Split Tour 2' FW + Split Tour 3' BW*
15. *Split Tour 1 BW + Split Tour 2' BW + Split Tour 3' FW*
16. *Split Tour 1 BW + Split Tour 2' BW + Split Tour 3' BW*

If we also consider the backwards version of the initial giant tour and the second best choice as VD, we will be able to generate 64 temporary solutions. These temporary solutions are the result of the multiple solution generation of the *Split-Approach* and will be used for further processing. One final remark, in our example we assume that all split tours have at least one VD inserted. Hence, we consider the forwards and backwards versions of the tours. However, if a split tour is feasible without the necessity of inserting a VD, the backwards version must not be considered because that tour is independent from the others. Thus, the forwards and backwards versions of that tour will result in the same final solution from a cost perspective. In Algorithm 3, we present the pseudo-code for the core algorithm of the *Split-Approach*.

3.6 Local search strategy (LS)

After generating multiple solutions for the SV class, the LV tours for all solutions are constructed by applying the same two steps as in the benchmark algorithm—sort all VDs in

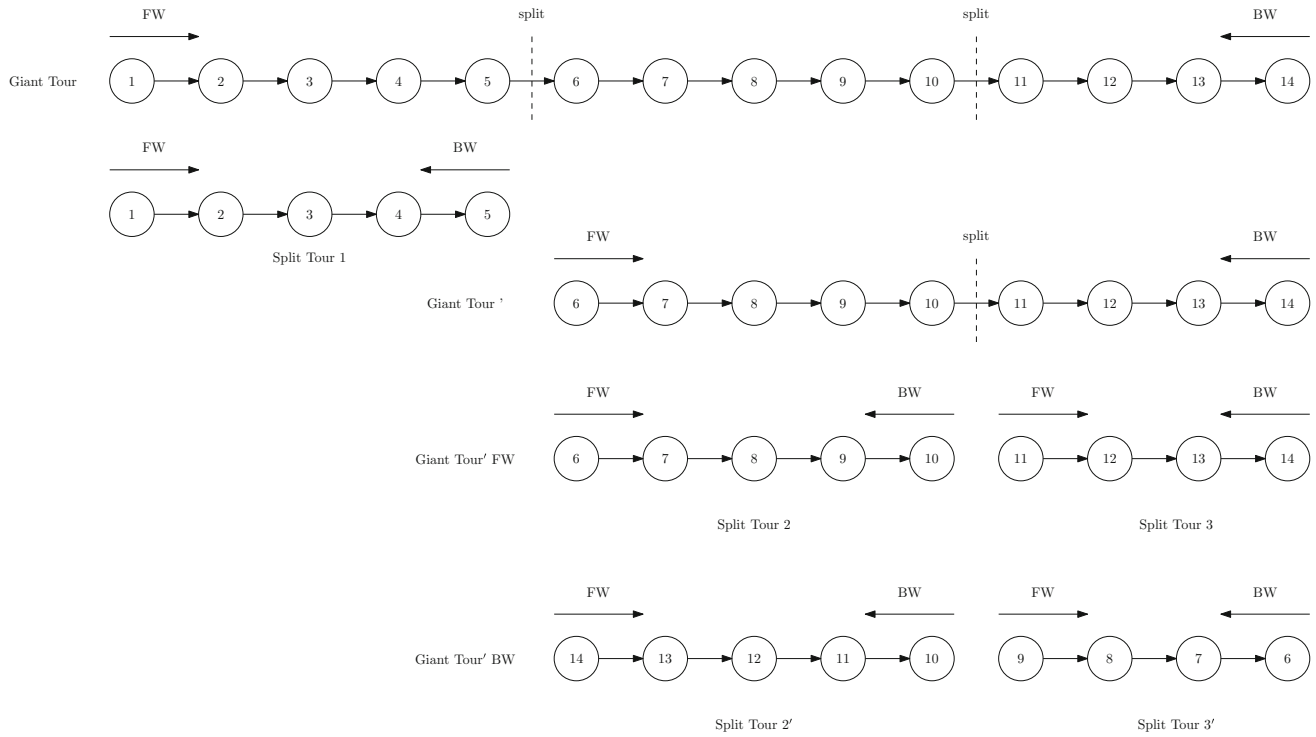


Fig. 5 *Split-Approach* multiple solutions

Algorithm 3: Core algorithm for the *Split-Approach*

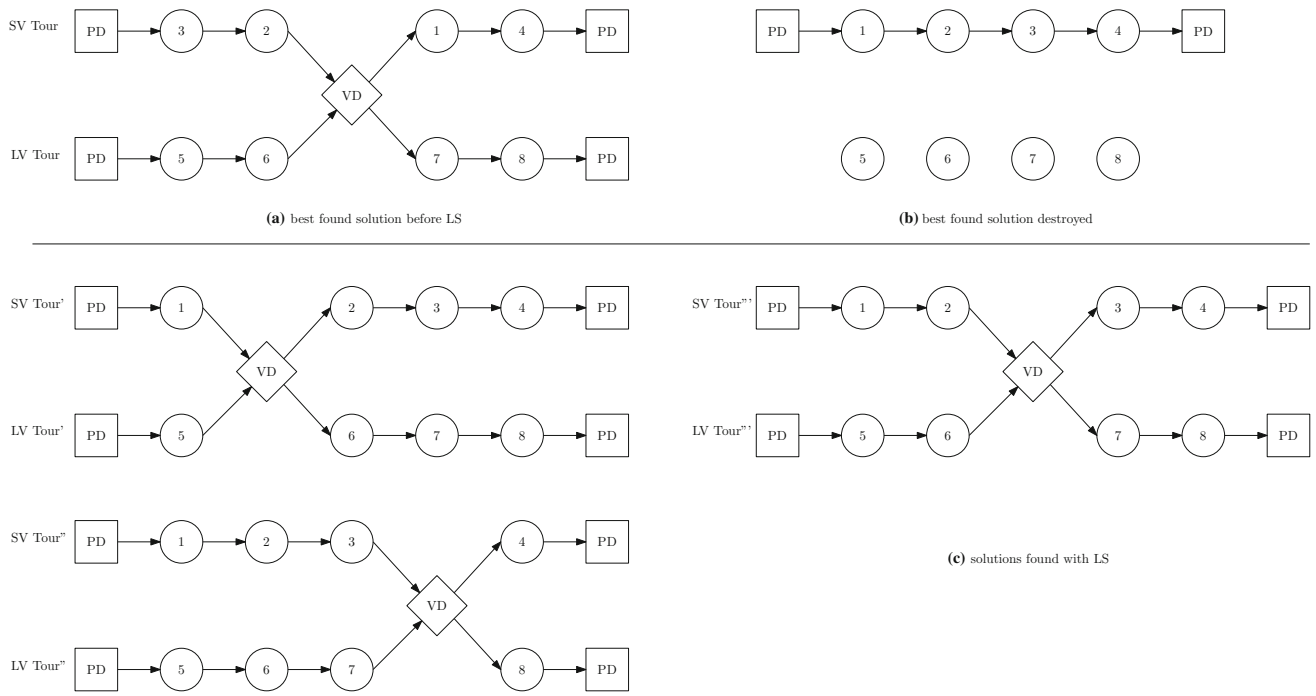
```
for first, second best VD or PD do
    for giant tour forwards and backwards do
        while giant tour is infeasible do
            while sub-tour is infeasible do
                for sub-tour forwards and backwards do
                    if a feasible solution is found then
                        insert VD/PD visits and save
                        solution in temp_sol;
                    else
                        continue;
                    end
                end
            end
        end
    end
    for all possible combinations in temp_sol do
        generate all SV tour combinations and save
        temporarily;
    end
end
```

chronological order, insert them into the LV tours and insert all the remaining type-1 customers into the SV or LV tours. As mentioned before, LV tours without synchronisation will be solved optimally and the final solution will be saved temporarily.

Even though we already improved our solution in the previous steps, we believe we can still further improve our algorithm by the use of local search (LS). Although the used local search procedure for our algorithm was originally developed for the *Linkage-Approach*, it is also applicable for the *Split-Approach*. The basic principle of local search is to explore the solution space within the neighbourhood of the current solution. Hence, we apply the local search procedure on the best found solution with minimum costs.

The applied local search procedure can be described as follows. We start by selecting the first SV tour with at least one transshipment customer (VD) and free all VDs from that tour and those from the associated LV tour. Afterwards, we destroy the remaining LV tour completely. In other words, all previously serviced customers (customers of type-1) of the LV tour are again unserviced. In the next step, we apply the matheuristics principle from the second improvement step to the selected SV tour and solve the routing optimally. Afterwards, we apply the core element of the *Split-Approach* to that tour. Contrary to the original core element of the *Split-Approach*, we consider now all possible VD insertion places that lead to a feasible tour. In other words, we search the solution space by generating again multiple solutions and choose the best solution among them. In Fig. 6, we provide a small illustrative example for a better understanding.

In our example, the best found solution has one SV and LV tour with one VD (see Fig. 6a). We now free the VD from both tours, destroy the LV tour so that all four (type-1) customers are unserviced again and solve the SV tour (TSP for customers 1,2,3 and 4) optimally. The result of these three steps is presented in Fig. 6b. After that, we apply the core-



(a) best found solution before LS

(b) best found solution destroyed

(c) solutions found with LS

Fig. 6 Local search procedure

algorithm of the *Split-Approach* and generate solutions for all three possible VD insertion places (between customers 1–2, 2–3 and 3–4; see Fig. 6c). If a superior solution is found, the current solution will be replaced and the LS procedure continues. That procedure will be repeated for all synchronized SV tours and the algorithm ends. The pseudo-code for the local search procedure is presented in Algorithm 4.

Algorithm 4: Core algorithm for the local search procedure

```
for all synchronized SV tours do
    free all VDs from the selected SV and associated LV tour;
    destroy associated LV tour and mark the customers unserviced;
    apply matheuristics to the selected SV tour;
    apply the core element of the Split-Approach and generate for all feasible solutions a final solution;
    search among all found solutions by the LS procedure;
    if a superior solution is found then
        accept current solution as best found solution;
    else
        continue;
    end
end
```

4 Computational results

The coding was done in programming language C and compiled with GNU GCC Compiler on a 4 * Intel® Core™ 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 TSP with the programming language Python (Version 2.7.6).

4.1 Set-up

To the best of our knowledge, Chen et al. [14] and Brandstätter and Reimann [5] are the sole available research papers for the LFVRP. The *Linkage-* and *Split-Approach* of Brandstätter and Reimann [5] provide superior results and therefore we consider them as benchmark for our analysis. Yet, we seize the opportunity to change/soften some of the limitations originally made by Chen et al. [14].

The limitations for change are the service/transshipment times, vehicle costs, average speed, depot location and set of test instances. As for the service and transshipment times, we will follow the approach from Brandstätter and Reimann [5] and set the transshipment time of a complete SV load to 15 minutes and the service times in relation to the demand of each customer. Furthermore, we will set the vehicle costs to European standards. For the LV, we consider a truck with a trailer with a 40t payload. The costs are as follows: €200 per day and vehicle, fuel consumption of 35l per 100 km—which results in €0.35/km and the drivers salary of €30 per hour. For the SV, we consider a cargo bike with costs of €2 per day and vehicle, €0.0/km (no fuel consumption) and the drivers salary of €20 per hour.

Table 1 Improvement strategy analysis

Strategy[a]					Linkage[b]			Split[b]			Stat. comp.[c]
No.	ME	MA	MS	LS	Solomon	Chen	Total	Solomon	Chen	Total	p value
1	0	0	0	0	1981	2375	2197	2013	2508	2285	**0.03**
2	0	0	0	1	1888	2341	2137	1970	2430	2223	**0.00**
3	0	0	1	0	1861	2281	2092	1954	2423	2212	**0.00**
4	0	0	1	1	1829	2279	2076	1926	2407	2190	**0.00**
5	0	1	0	0	1941	2347	2164	1960	2414	2209	0.37
6	0	1	0	1	1879	2306	2114	1919	2377	2171	0.19
7	0	1	1	0	1856	2254	2075	1929	2378	2176	**0.04**
8	0	1	1	1	1818	2259	2060	1915	2362	2161	**0.03**
9	1	0	0	0	1958	2213	2098	1995	2416	2227	**0.00**
10	1	0	0	1	1833	2176	2022	1950	2377	2185	**0.00**
11	1	0	1	0	1816	2147	1998	1944	2377	2182	**0.00**
12	1	0	1	1	1782	2119	1967	1936	2356	2167	**0.00**
13	1	1	0	0	1939	2200	2082	1960	2414	2209	**0.01**
14	1	1	0	1	1840	2157	2014	1919	2377	2171	**0.00**
15	1	1	1	0	1799	2122	1976	1929	2381	2178	**0.00**
16	1	1	1	1	1742	2103	1941	1915	2367	2164	**0.00**

[a]Improvement strategies: metaheuristic (ME), matheuristic (MA), multiple solutions (MS) and local search (LS). 1 indicates an enabled strategy, and 0 a disabled strategy
[b]Average cost values over total instance set
[c]Results of WSRT for the comparison of total costs. For values in bold: p value $< 0.05 \rightarrow H_0$ rejected

For this paper, we decided to introduce different average speed values for the vehicle classes. We maintain the average speed of 40 km/h for the LV but needed to reduce the average speed for the SV due to the use of eco-friendly cargo bikes. Hence, we set the average speed for the SV to 15 km/h.

Furthermore, wee keep the SV capacity constant with 100 units and therefore not all original test instances were eligible. Hence, we consider only the first 11 (A) test instances of Chen et al. [14] and extend the set with test instances of the well-known VRPTW test instance set proposed by Solomon [44]. As we do not consider time windows, it does not make sense to use all 56 test instances. Therefore, we decided do use just the 101 dataset with 100 customers (for classes C, R and RC). In addition, we also divided a single set into 25, 50 and 100 customers. Thus, we gain 9 additional test instances which will result in a total of 20 test instances for our analysis. For the remainder of the paper, we will refer to the 11 (A) test instances as the Chen instances and the 9 C,R and RC test instances as the Solomon instances.

The customer distribution of type-1 and type-2 customers is the next limitation we changed. We set the customer distribution to 1:3 for the Solomon instances, and we left the customer distribution of the Chen instances as they were (4 type-1 customers for each instance). In other words, the majority of the customers (75% customers of type-2 for the Solomon instances) must be served by the SV class, whereas 25% can be chosen as a VD and served by both vehicle classes. Finally, we wanted to simulate a PD outside the city borders with the coordinates of 0/0.

4.2 Data analysis

Before we have a look at the best found results, we want to analyse the impact of the different improvement strategies along with their dependencies. Therefore, we implemented the four improvement strategies to be independent, so we can enable or disable them separately. Table 1 shows the 16 strategy combinations (four strategies which are either enabled or disabled) for our analysis.

The first column of Table 1 indicates the combination number, whereas the following four columns show which improvement strategy is enabled (1) or disabled (0). For example, Strategy 10 has the metaheuristic enabled (1), matheuristic (0) and multiple solutions (0) disabled and local search enabled (1). The next six columns show the average cost results (divided into Solomon, Chen and Total results) for the *Linkage*- and *Split-Approach* gained after 10 repetition cycles. In the last column, we present the results for the statistical comparison. We compared the average total costs of both approaches by using the Wilcoxon signed rank test (WSRT see [48]) with the Null Hypothesis H_0 if both approaches have the same solution quality and the Alternative Hypothesis H_1 if the difference in solution quality is significant.

What we can observe is that the *Linkage-Approach* outperforms the *Split-Approach* regardless of the strategy combination. The advantage is substantial for almost every strategy combination except for two—strategy combination 5 and 6. Here, both approaches provide similar results in terms of

Table 2 Improvement strategy ranking

	Linkage-Approach		Split-Approach	
Rank	Total cost	Strategy[a]	Total cost	Strategy[a]
1	1941	16	2161	8
2	1967	12	2164	16
3	1976	15	2167	12
4	1998	11	2171	6
5	2014	14	2171	14
6	2022	10	2176	7
7	2060	8	2178	15
8	2075	7	2182	11
9	2076	4	2185	10
10	2082	13	2190	4
11	2092	3	2209	5
12	2098	9	2209	13
13	2114	6	2212	3
14	2137	2	2223	2
15	2164	5	2227	9
16	2197	1	2285	1
Min.	1941	16	2161	8
Max.	2197	1	2285	1
Range	256	n.a.	124	n.a.

[a]Strategy no. from first column of Table 1

Table 3 Statistical analysis *Linkage-Approach*

Strategy set	Comparison	p value[a]
None	ME versus no ME	**0.00**
	MA versus no MA	**0.00**
	MS versus no MS	**0.00**
	LS versus no LS	**0.00**
ME = 1	MA versus no MA	**0.03**
	MS versus no MS	**0.00**
	LS versus no LS	**0.00**
MA = 1	ME versus no ME	**0.00**
	MS versus no MS	**0.00**
	LS versus no LS	**0.00**
MS = 1	ME versus no ME	**0.00**
	MA versus no MA	**0.00**
	LS versus no LS	**0.00**
LS = 1	ME versus no ME	**0.00**
	MA versus no MA	**0.00**
	MS versus no MS	**0.00**

[a]p value $< 0.05 \rightarrow H_0$ rejected

solution quality. As expected, the biggest advantage—more than 10%—can be realised if all four strategies are enabled.

Table 2 shows the ranking of the different strategies. We can observe that the best results (Rank 1–6) for the *Linkage-Approach* are achieved with an enabled metaheuristic. However, the best results (Rank 1–7) for the *Split-Approach* are mixed with an enabled matheuristic. The reason for that can be found in the first two improvement strategies.

The matheuristic provides the optimal solution for the giant tour. Consequently, the first step with the metaheuristic approach becomes obsolete. That can be seen if we compare the total costs of the *Split-Approach* (which are almost equal) for ranks 1–2, 4–5, 6–7 and 11–12. Finally, 14 strategy combinations of the *Linkage-Approach* are superior to the best found result of the *Split-Approach*. Therefore, we decided to skip the *Split-Approach* for further analysis and proceed only with the *Linkage-Approach*.

With the previous analysis, we gained an interesting insight. We found the best results for the *Linkage-Approach* with all improvement strategies enabled. However, to prove the impact of each improvement strategy on an individual level, we needed an additional analysis. Hence, we conducted a multi-stage statistical analysis. In the first stage, we compared the results of the improvement strategies on their own. In other words, we compared the total costs with enabled and disabled improvement strategy over the total instance set.

Afterwards, we enabled each improvement strategy separately and compared the remaining strategies again (second stage), e.g. enable (1) MS as prerequisite and compare MA versus no MA, MS versus no MS and LS versus no LS. In the third/fourth stage, we enabled two/three improvement strategies as prerequisite compared the results of the remaining strategies accordingly. Table 3 shows the results of the first two stages, and the remaining results can be found in Table 6 of the "Appendix".

Table 3 shows that all strategy combinations yield a significant contribution to the final result. The only exception (which can be found in Table 7 of the "Appendix") is the matheuristic (MA) strategy with enabled metaheuristic (ME) and local search (LS) strategy. With that constellation, we were not able to provide a significant result contribution. Yet, we can conclude that the impact of all four improvement strategies is substantial.

In Table 4, we present the final results for the *Linkage-Approach* gained with strategy 16 (all improvement strategies enabled). We were able to reduce the total costs—in comparison with the benchmark results presented in Table 7 (see "Appendix")—for the Solomon instance set by 7.8% and the Chen instance set by 9.2% which results in a total improvement by 8.6%. Furthermore, no. of vehicles, driven distance and needed time were reduced for the SV class and slightly increased for the LV class. No. of required reloads (at the PD and VDs) and the required cpu time increased as well. For the sake of completeness, we present the final results for the Spit-Approach in Table 8 of the "Appendix".

Finally, we present the best found results regardless of the chosen approach and strategy in Table 5. As expected,

Table 4 Final results for *Linkage-Approach*

Instance	Cost	No. SV	No. LV	Dist. SV	Time. SV	Dist. LV	Time, LV	Reload	CPU Time[b]
C101_25	850	1	1	226.52	660.04	196.94	331.75	3	2.422
C101_50	1245	2	1	448.44	1317.36	254.24	391.22	4	3.002
C101_100	2699	4	1	1156.03	3522.23	519.76	731.12	9	4.893
R101_25	846	3	0	613.05	1679.71	0.00	0.00	0	2.908
R101_50	1502	2	1	608.71	1722.04	266.49	458.41	4	3.030
R101_100	2324	3	1	964.62	2772.72	369.42	803.08	9	5.047
RC101_25	1071	1	1	258.13	764.34	216.53	547.78	4	2.640
RC101_50	1802	3	1	761.90	2172.27	299.05	539.93	5	2.985
RC101_100	2786	4	1	1245.46	3638.75	532.14	762.59	10	4.975
A32	1490	2	1	689.71	1895.32	291.71	314.81	2	2.694
A34	1592	3	1	802.10	2190.94	253.98	268.98	1	2.998
A38	1658	3	1	821.43	2250.77	278.69	304.99	2	3.028
A45	1979	4	1	1058.68	2899.05	275.55	299.45	2	2.991
A46	1787	3	1	867.54	2395.15	270.09	385.67	3	2.995
A60	2202	4	1	1206.38	3335.01	267.15	310.15	4	3.002
A61	2157	3	1	993.28	2787.15	340.29	584.33	6	2.963
A64	2175	4	1	1181.48	3270.30	227.19	336.75	4	3.018
A65	2277	4	1	1215.49	3359.80	321.75	369.25	4	3.038
A69	2320	4	1	1304.87	3601.54	263.46	291.46	2	2.962
A80	2768	5	1	1624.61	4458.19	253.74	320.22	4	3.093
Average Solomon[a]	1680 (7.8%)	2.56 (11.5%)	0.89 (11.1%)	698.10 (10.9%)	2027.72 (10.7%)	294.95 (16.0%)	507.32 (−6.3%)	5.33 (−6.7%)	3.545 (0.150)[c]
Average Chen[a]	2037 (9.2%)	3.55 (22.0%)	1.00 (−22.2%)	1069.60 (18.9%)	2949.38 (18.3%)	276.69 (−31.3%)	344.19 (−33.6%)	3.09 (−41.7%)	2.980 (0.144)[c]
Average Total[a]	1876 (8.6%)	3.10 (18.4%)	0.95 (−5.6%)	902.42 (16.3%)	2534.63 (15.7%)	284.91 (−4.0%)	417.60 (−17.1%)	4.10 (−18.8%)	3.234 (0.147)[c]

[a]Deviation from benchmark values in parenthesis.
[b,c]CPU time in seconds, benchmark values in parenthesis

Table 5 Best found results from both approaches

Instance	Cost	CPU	Approach
C101_25	850	0.040	Linkage
C101_50	1239	0.369	Linkage
C101_100	2448	1.992	Linkage
R101_25	846	2.374	Linkage
R101_50	1474	2.978	Split
R101_100	2281	18.658	Split
RC101_25	1049	0.070	Split
RC101_50	1757	3.072	Linkage
RC101_100	2786	4.975	Linkage
A32	1488	2.202	Linkage
A34	1550	2.999	Linkage
A38	1658	3.028	Linkage
A45	1900	0.229	Linkage
A46	1787	2.994	Linkage
A60	2202	3.002	Linkage
A61	2157	2.963	Linkage
A64	2146	3.007	Linkage
A65	2214	3.002	Linkage
A69	2300	3.010	Linkage
A80	2768	3.093	Linkage
Average Solomon	1637	3.836	n.a.
Average Chen	2015	2.684	n.a.
Average Total	1845	3.203	n.a.

the majority of the best results are provided by *Linkage-Approach*. Only for three instances—R101_50, R101_100 and RC101_25—the *Split-Approach* provided better results. In summary, we provided evidence that the *Linkage-Approach* is the superior approach and can be henceforth considered as new benchmark for the LFVRP.

5 Conclusion

City logistics are facing new challenges these days. Some of these challenges are rising land prices and customer demands, cities with limited space and increasing traffic or additional environmental regulations (e.g. to reduce particulate matter pollution). Current VRP variants address some of these challenges but do not cover all of them. Thus, new variants with multiple challenges arise and one of them is the LFVRP.

The LFVRP uses a heterogeneous fleet of vehicles (small and large) to serve two types of customers. Customers of type-1 have enough space and can be accessed by both vehicle classes, whereas customers of type-2 can only be accessed by the SV class due to limited space. The main characteristic, that distinguishes the LFVRP from other VRP variants, is the synchronisation between vehicles. Synchronisation takes place if a small and large vehicle meet at a customer with enough space and perform a reload operation. In our first paper, we formally defined the problem and introduced two promising heuristics called the *Linkage-* and *Split-Approach*. We provided evidence that the synchronisation of vehicles reduces significantly the no. of (small) vehicles and therefore the total required costs.

With this paper, we improved the two previously proposed approaches by using the benefits of *metaheuristics*, *matheuristics*, generating *multiple solutions* and *local search*. In the first step, we focused on the improvement of the starting solution which is in its core a VRP- (for the *Linkage-Approach*) and a TSP-solution (for the *Split-Approach*), respectively. Although there are several ways to improve a VRP- or TSP solution, we decided to use the *metaheuristic* ACO due to earlier positive results. In the second step, we applied a matheuristic strategy, specifically the decomposition method where single vehicle tours are extracted from the solution and solved optimally. For the *Linkage-Approach*, we solved all SV tours, and for the *Split-Approach* the giant tour. Furthermore, the matheuristic strategy is also applied to all LV tours without synchronisation (no VDs). In the next step, we generated multiple solutions by using different construction strategies. For the *Linkage-Approach*, we adopted the most common bin packing strategies. The forwards and backwards consideration of the tours along with the best and second best VD positions were used for both approaches. In the final step, we applied a local search strategy by using a destroy-and-repair approach.

For our numerical analysis, we extended the previous provided test instances and altered some parameters, e.g. different (average) speed for the vehicle classes, capacity for the SV (as we use cargo bikes), more realistic transshipment times, and we set the costs to a European standard. In the first part of our analysis, we investigated the impact of different strategy combinations. In total, we analysed 16 strategy combinations and we were able to significantly improve both our approaches. As expected, the best results for the *Linkage-Approach* were achieved by enabling all four strategies and for the *Split-Approach* by enabling only three strategies (*matheuristics*, *multiple solutions* and *local search*). Even though both approaches were improved significantly, the *Linkage-Approach* clearly outperformed the *Split-Approach*. Hence, we proceeded with the statistical analysis only for the *Linkage-Approach* where we analysed the impact of each individual strategy and strategy combination. The contribution to the final result was substantial for almost all strategy combinations. Overall, we were able to reach an average improvement of around 9% over previous approaches.

This paper serves as solid foundation for further investigations on the LFVRP. The results clearly show that the LFVRP

has a huge potential—especially the multiple solutions strategy. Another interesting direction will be the extension or further development of the problem in terms of the current set-up (e.g. usage of cargo bikes as SV). For instance, if cargo bikes are used, it is unlikely that they will also start from the PD if it is located far outside of the city. In that case, it is more likely that they will start their tours from somewhere within the city. If that is the case, they must first meet with a LV to pick up their load so that they can start servicing the customers. Nonetheless, the LFVRP has huge potential and is worth to be investigated further.

Acknowledgements Open access funding provided by University of Graz.

Compliance with ethical standards

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

Appendix

See Tables 6, 7 and 8.

Table 6 Statistical analysis *Linkage-Approach* (Table 3 continued)

Strategy set	Comparison	p value[a]
ME = 1 and MA = 1	MS versus no MS	**0.00**
	LS versus no LS	**0.00**
ME = 1 and MS = 1	MA versus no MA	**0.03**
	LS versus no LS	**0.00**
ME = 1 and LS = 1	MA versus no MA	0.05
	MS versus no MS	**0.00**
MA = 1 and MS = 1	ME versus no ME	**0.00**
	LS versus no LS	**0.00**
MA = 1 and LS = 1	ME versus no ME	**0.00**
	MS versus no MS	**0.00**
MS = 1 and LS = 1	ME versus no ME	**0.00**
	MA versus no MA	**0.00**
ME = 1, MA = 1 and MS = 1	LS versus no LS	**0.01**
ME = 1, MA = 1 and LS = 1	MS versus no MS	**0.00**
ME = 1, MS = 1 and LS = 1	MA versus no MA	**0.01**
MA = 1, MS = 1 and LS = 1	ME versus no ME	**0.00**

[a] p value $< 0.05 \rightarrow H_0$ rejected

Table 7 Benchmark results for *Linkage-Approach*

Instance	Cost	No. SV	No. LV	Dist. SV	Time. SV	Dist. LV	Time, LV	Reload	CPU time
C101_25	915	1	1	272.20	782.86	196.94	336.31	3	0.008
C101_50	1347	2	1	509.21	1457.90	249.65	436.13	4	0.052
C101_100	2777	4	1	1145.72	3503.55	646.69	787.69	9	0.367
R101_25	997	2	1	427.17	1176.01	174.03	191.93	1	0.008
R101_50	1562	3	1	739.11	2059.85	263.02	311.92	3	0.054
R101_100	2568	4	1	1249.87	3557.41	411.48	582.52	8	0.427
RC101_25	1125	2	1	435.05	1254.50	219.36	288.89	3	0.006
RC101_50	1831	3	1	789.39	2235.04	299.05	537.47	5	0.046
RC101_100	3274	5	1	1484.00	4404.26	698.81	823.61	9	0.379
A32	1590	3	1	835.77	2277.72	214.99	226.99	1	0.017
A34	1618	3	1	816.40	2229.07	260.35	275.35	1	0.021
A38	1819	5	0	1335.23	3618.52	0.00	0.00	0	0.034
A45	2112	6	0	1548.50	4199.23	0.00	0.00	0	0.056
A46	1990	4	1	1097.20	2997.47	246.02	263.02	1	0.065
A60	2728	5	1	1570.12	4296.68	312.80	346.60	3	0.146
A61	2356	4	1	1240.47	3440.32	267.05	446.21	5	0.237
A64	2332	4	1	1304.20	3598.56	232.39	324.00	4	0.182
A65	2508	5	1	1435.55	3937.03	265.89	315.30	3	0.185
A69	2580	5	1	1491.06	4088.17	263.46	311.01	3	0.262
A80	3051	6	1	1831.70	5010.44	254.96	325.42	3	0.382
Average Solomon	1822	2.89	1.00	783.52	2270.15	351.00	477.39	5.00	0.150
Average Chen	2244	4.55	0.82	1318.75	3608.47	210.72	257.63	2.18	0.144
Average Total	2054	3.80	0.90	1077.90	3006.23	273.85	356.52	3.45	0.147

Table 8 Final results for *Split-Approach*

Instance	Cost	No. SV	No. LV	Dist. SV	Time. SV	Dist. LV	Time, LV	Reload	CPU time
C101_25	970	2	1	378.29	1059.78	196.60	223.60	2	2.427
C101_50	1393	4	1	638.57	1839.19	203.78	258.78	4	3.980
C101_100	3201	9	1	1747.61	4865.30	430.67	532.90	5	81.156
R101_25	962	2	1	424.02	1167.83	148.47	162.57	1	2.998
R101_50	1582	4	1	810.69	2248.14	203.62	237.92	3	3.963
R101_100	2362	6	1	1216.90	3431.37	266.23	454.51	7	16.594
RC101_25	1109	2	1	425.22	1206.91	216.53	301.05	3	2.748
RC101_50	2426	7	1	1374.94	3770.50	278.73	305.73	2	10.784
RC101_100	3231	9	1	1764.68	4926.89	433.92	530.86	7	112.017
A32	1618	5	0	1190.88	3216.68	0.00	0.00	0	3.914
A34	1750	5	0	1286.74	3480.70	0.00	0.00	0	2.924
A38	1767	4	1	981.33	2667.77	196.80	207.70	1	2.593
A45	2088	5	1	1263.65	3434.94	138.38	149.58	1	2.975
A46	2006	4	1	1083.00	2959.80	246.02	309.14	2	2.894
A60	2769	7	1	1721.69	4691.58	178.35	195.75	2	4.903
A61	2539	7	1	1509.79	4133.90	217.55	242.35	3	5.341
A64	2536	6	1	1494.72	4096.82	227.19	261.09	3	2.932
A65	2847	7	1	1698.25	4635.47	265.89	295.69	3	3.936
A69	2886	7	1	1753.72	4796.49	227.92	258.29	3	4.658
A80	3175	8	1	1952.14	5312.42	249.75	287.81	3	12.044
Average Solomon	1915	5.00	1.00	975.66	2723.99	264.28	334.21	3.78	26.296
Average Chen	2362	5.91	0.82	1448.72	3947.87	177.08	200.67	1.91	4.465
Average Total	2161	5.50	0.90	1235.84	3397.12	216.32	260.77	2.75	14.289

References

1. Archetti, C., Speranza, M.G.: A survey on matheuristics for routing problems. EURO J. Comput. Optim. **2**, 223–246 (2014)
2. Baker, B., Ayechew, M.: A genetic algorithm for the vehicle routing problem. Comput. Oper. Res. **30**, 787–800 (2003)
3. Baldacci, R., Battarra, M., Vigo, D.: Routing a Heterogeneous Fleet of Vehicles. In: Golden, B., Raghavan, S., Wasil, E. (eds.) The Vehicle Routing Problem: Latest Advances and New Challenges, pp. 3–27. Springer US, Boston (2008)
4. Boschetti, M.A., Maniezzo, V., Roffilli, M., Bolufé Röhler, A.: Matheuristics: optimization, simulation and control. In: Hutchison, D., Kanade, T., Kittler, J., Kleinberg, J.M., Mattern, F., Mitchell, J.C., Naor, M., Nierstrasz, O., Pandu Rangan, C., Steffen, B., Sudan, M., Terzopoulos, D., Tygar, D., Vardi, M.Y., Weikum, G., Blesa, M.J., Blum, C., Di Gaspero, L., Roli, A., Sampels, M., Schaerf, A. (eds.) Hybrid Metaheuristics. volume 5818 of Lecture Notes in Computer Science, pp. 171–177. Springer, Berlin (2009)
5. Brandstätter, C., Reimann, M.: The line-haul feeder vehicle routing problem: Mathematical model formulation and heuristic approaches. Eur. J. Oper. Res. **270**(1), 157–170 (2018). https://doi.org/10.1016/j.ejor.2018.03.014
6. Bräysy, O., Gendreau, M.: Vehicle routing problem with time windows, Part I: route construction and local search algorithms. Transp. Sci. **39**, 104–118 (2005)
7. Bräysy, O., Gendreau, M.: Vehicle routing problem with time windows, Part II: metaheuristics. Transp. Sci. **39**, 119–139 (2005)
8. Cattaruzza, D., Absi, N., Feillet, D.: Vehicle routing problems with multiple trips. 4OR-A Q. J. Oper. Res. **14**, 223–259 (2016)
9. Cattaruzza, D., Absi, N., Feillet, D., Vidal, T.: A memetic algorithm for the multi trip vehicle routing problem. Eur. J. Oper. Res. **236**, 833–848 (2014)
10. Chao, I.M., Golden, B., Wasil, E.: A computational study of a new heuristic for the site-dependent vehicle routing problem. INFOR Inf. Syst. Oper. Res. **37**, 319–336 (2016)
11. Cheikh, M., Ratli, M., Mkaouar, O., Jarboui, B.: A variable neighborhood search algorithm for the vehicle routing problem with multiple trips. Electron. Notes Discrete Math. **47**, 277–284 (2015)
12. Chen, H.K.: Issues for the linehaul-feeder vehicle routing problem with virtual depots and time windows. J. East. Asia Soc. Transp. Stud. **11**, 678–692 (2015)
13. Chen, H.-K., Chou, H.-W., Hsu, C.-Y.: The linehaul-feeder vehicle routing problem with virtual depots and time windows. In: Mathematical Problems in Engineering, vol. 2011 (2011)
14. Chen, H.K., Chou, H.W., Hsueh, C.F., Ho, T.Y.: The linehaul-feeder vehicle routing problem with virtual depots. IEEE Trans. Autom. Sci. Eng. **8**, 694–704 (2011)
15. Chen, H.K., Wang, H.: A two-stage algorithm for the extended linehaul-feeder vehicle routing problem with time windows. Int. J. Shipp. Transp. Logist. **4**, 339–356 (2012)
16. Cordeau, J.F., Gendreau, M., Laporte, G., Potvin, J.Y., Semet, F.: A guide to vehicle routing heuristics. J. Oper. Res. Soc. **53**, 512–522 (2002)
17. Cordeau, J.F., Laporte, G.: A tabu search algorithm for the site dependent vehicle routing problem with time windows. INFOR Inf. Syst. Oper. Res. **39**, 292–298 (2016)

18. Dantzig, G., Fulkerson, R., Hohnson, S.: Solution of a large-scale traveling salesman problem. J. Oper. Res. Soc. Am. **2**, 393–410 (1954)
19. Delorme, M., Iori, M., Martello, S.: Bin packing and cutting stock problems: mathematical models and exact algorithms. Eur. J. Oper. Res. **255**, 1–20 (2016)
20. Doerner, K.F., Schmid, V.: Survey: matheuristics for rich vehicle routing problems. In: Blesa, M.J. (ed.) Hybrid Metaheuristics. Volume 6373 of Lecture Notes in Computer Science, pp. 206–221. Springer, Berlin (2010)
21. Dorigo, M., Blum, C.: Ant colony optimization theory: a survey. Theor. Comput. Sci. **344**, 243–278 (2005)
22. European Environment Agency. Air quality in Europe 2016 report. https://doi.org/10.2800/80982. https://www.eea.europa.eu/publications/air-quality-in-europe-2016/ (2016). Accessed 21 Aug 2017
23. François, V., Arda, Y., Crama, Y., Laporte, G.: Large neighborhood search for multi-trip vehicle routing. Eur. J. Oper. Res. **255**, 422–441 (2016)
24. Glover, F.: Tabu search-Part I. INFORMS J. Comput. **1**, 190–206 (1989)
25. Glover, F.: Tabu search-Part II. INFORMS J. Comput. **2**, 4–32 (1990)
26. Golden, B., Assad, A., Levy, L., Gheysens, F.: The fleet size and mix vehicle routing problem. Comput. Oper. Res. **11**, 49–66 (1984)
27. Gurobi Optimization, Inc. Gurobi optimizer reference manual. http://www.gurobi.com (2016)
28. Johnson, D.S.: Fast algorithms for bin packing. J. Comput. Syst. Sci. **8**, 272–314 (1974)
29. Kirkpatrick, S., Gelatt, C., Vecchi, M.: Optimization by simulated annealing. Science **220**, 671–680 (1983)
30. Koç, Ç., Bektaş, T., Jabali, O., Laporte, G.: Thirty years of heterogeneous vehicle routing. Eur. J. Oper. Res. **249**, 1–21 (2016)
31. Kritikos, M.N., Ioannou, G.: The heterogeneous fleet vehicle routing problem with overloads and time windows. Int. J. Prod. Econ. **144**, 68–75 (2013)
32. Laporte, G.: The traveling salesman problem—an overview of exact and approximate algorithms. Eur. J. Oper. Res. **59**, 231–247 (1992)
33. Laporte, G.: Fifty years of vehicle routing. Transp. Sci. **43**, 408–416 (2009)
34. Laporte, G., Ropke, S., Vidal, T.: Chapter 4: Heuristics for the Vehicle Routing Problem. In: Toth, P., Vigo, D. (eds.), Vehicle Routing, pp. 87–116 (2014)
35. Liu, F.H., Shen, S.Y.: The fleet size and mix vehicle routing problem with time windows. J. Oper. Res. Soc. **50**, 721 (1999)
36. Penna, P.H.V., Subramanian, A., Ochi, L.S.: An iterated local search heuristic for the heterogeneous fleet vehicle routing problem. J. Heuristics **19**, 201–232 (2013)
37. Pisinger, D., Ropke, S.: A general heuristic for vehicle routing problems. Comput. Oper. Res. **34**(8), 2403–2435 (2007)
38. Pisinger, D., Ropke, S.: Large neighborhood search. In: Gendreau, M., Potvin, J.Y. (eds.) Handbook of Metaheuristics, pp. 399–419. Springer US, Boston (2010)
39. Prins, C.: A simple and effective evolutionary algorithm for the vehicle routing problem. Comput. Oper. Res. **31**, 1985–2002 (2004)
40. Reimann, M., Doerner, K., Hartl, R.F.: D-ants: savings based ants divide and conquer the vehicle routing problem. Comput. Oper. Res. **31**, 563–591 (2004)
41. Ropke, S., Pisinger, D.: An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transp. Sci. **40**, 455–472 (2006)
42. Ropke, S., Pisinger, D.: A unified heuristic for a large class of vehicle routing problems with backhauls. Eur. J. Oper. Res. **171**, 750–775 (2006)
43. Senarclens de Grancy, G.: Applied metaheuristics for logistical challenges in congested urban areas: three essays: Dissertation. Universität Graz, Institut für Statistik und Operations Research, Graz (2015)
44. Solomon, M.M.: Algorithms for the vehicle routing and scheduling problems with time window constraints. Oper. Res. **35**, 254–265 (1987)
45. Subramanian, A., Penna, P.H.V., Uchoa, E., Ochi, L.S.: A hybrid algorithm for the heterogeneous fleet vehicle routing problem. Eur. J. Oper. Res. **221**, 285–295 (2012)
46. United Nations, Department of Economic and Social Affairs, Population Division. World Urbanization Prospects: The 2014 Revision, custom data acquired via website. https://esa.un.org/unpd/wup/DataQuery/ (2014). Accessed 26 June 2017
47. van Breedam, A.: Improvement heuristics for the vehicle routing problem based on simulated annealing. Eur. J. Oper. Res. **86**, 480–490 (1995)
48. Wilcoxon, F.: Individual comparisons by ranking methods. Biom. Bull. **1**, 80–83 (1945)
49. Yao, A.C.C.: New algorithms for bin packing. J. ACM **27**, 207–227 (1980)