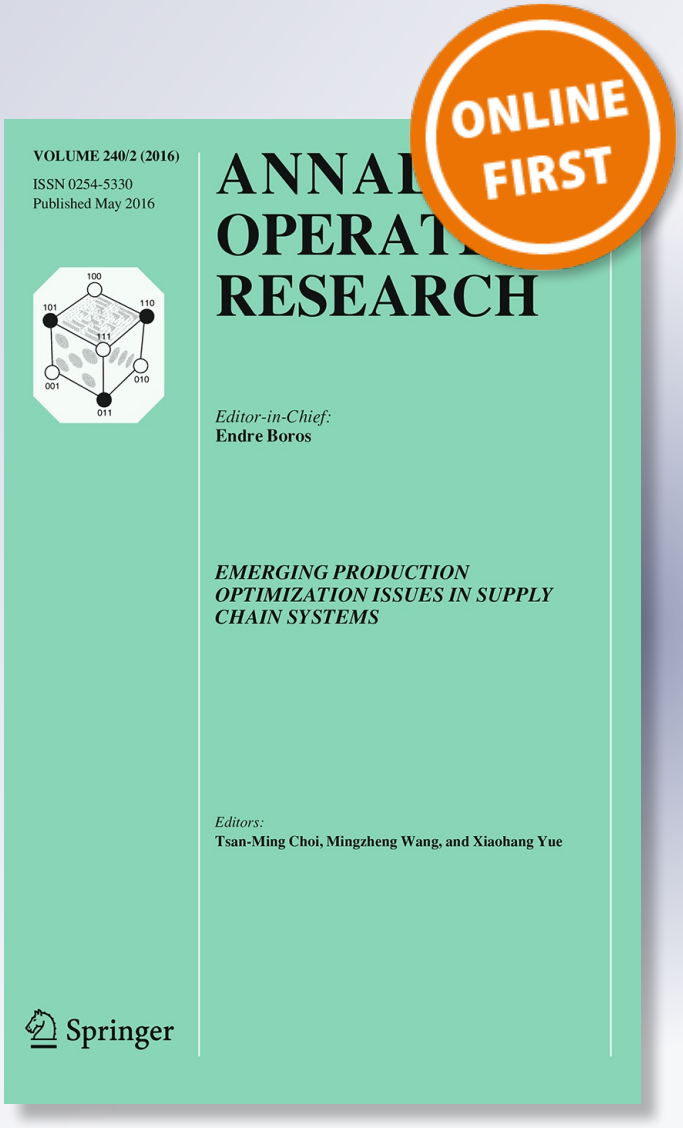
A cluster-first route-second approach for the swap body vehicle routing problem

Juan José Miranda-Bront, Brian Curcio, Isabel Méndez-Díaz, Agustín Montero, Federico Pousa & Paula Zabala

Annals of Operations Research

ISSN 0254-5330

Ann Oper Res
DOI 10.1007/s10479-016-2233-1



Springer

Ann Oper Res
DOI 10.1007/s10479-016-2233-1

S.I.: LOT

A cluster-first route-second approach for the swap body vehicle routing problem

Juan José Miranda-Bront[1,2] · **Brian Curcio**[1,2] · **Isabel Méndez-Díaz**[1] · **Agustín Montero**[1,2] · **Federico Pousa**[1,2] · **Paula Zabala**[1,2]

Abstract The swap body vehicle routing problem (SB-VRP) is a generalization of the classical vehicle routing problem where a particular structure as well as several operational aspects for the trucks composing the fleet are considered. This research has been motivated by the VeRoLog Solver Challenge 2014, organized together by VeRoLog and PTV group, aiming to motivate the study of real-world logistic problems. A truck can carry either only one swap body or, in addition, an extra trailer with an extra swap body. For the latter, special depots, called *swap locations*, can be used to drop and pickup the swap bodies. These operations may affect the feasibility and the cost of a route, and therefore the overall operational cost. In this paper, we propose a cluster-first route-second heuristic for the SB-VRP. Computational experiments are conducted over the benchmark instances proposed for the competition, simulating a practical environment by considering limited resources and execution time. The results obtained are of very good quality, where our approach ended as runner-up in the final set of instances and performs similarly to the other algorithms in the remaining cases, showing its potential to be applied in practice.

✉ Juan José Miranda-Bront
jmiranda@dc.uba.ar

Brian Curcio
bcurcio@dc.uba.ar

Isabel Méndez-Díaz
imendez@dc.uba.ar

Agustín Montero
aimontero@dc.uba.ar

Federico Pousa
fpousa@dc.uba.ar

Paula Zabala
pzabala@dc.uba.ar

1 Departamento de Computación, FCEyN, Universidad de Buenos Aires, Pabellón I, Ciudad Universitaria, C1428EGA Ciudad Autónoma de Buenos Aires, Argentina

2 Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), Buenos Aires, Argentina

Published online: 28 May 2016

Keywords Swap-body VRP · Iterated local search · GRASP · Cluster-first route-second

1 Introduction

Vehicle routing problem (VRP) is one of the most important and studied problems in logistic and transportation literature. Many variants covering different aspects of real-world situations define problems with particular characteristics. Toth et al. (2014) and Golden et al. (2008) provide a very detailed and extensive survey including practical applications of the VRP as well as the proper algorithms to tackle them.

The EURO Working Group on vehicle routing and logistics (VeRoLog), in collaboration with PTV Group, proposed the VeRoLog Solver Challenge 2014, aimed to develop and compare solution approaches of participants from the academic and professional sector for a specific variant of the VRP, called *swap body VRP* (SB-VRP). The SB-VRP is a generalization of the VRP that considers a homogeneous fleet which consists of trucks, semi-trailers and swap bodies, as well as special locations, called *swap locations*. The swap locations represent special depots where the swap bodies can be coupled and decoupled. In addition, some customers can only be visited by trucks having a particular configuration, i.e., carrying only one swap body.

Problems having a similar structure as the SB-VRP have been considered in the literature, which are reviewed in the next section. Regarding the SB-VRP, several algorithms have been proposed within the context of the competition. To the best of our knowledge, the only published approaches are the one by Huber and Geiger (2014), Lum et al. (2015) and recently Absi et al. (2015). Huber and Geiger (2014) describe the heuristic developed for the competition and report the results obtained over the test instances provided by the organizers. They consider an iterated local search (ILS). An initial solution is constructed randomly, and then a sequence of intra and inter tour local search operators are applied. Whenever the current solution cannot be improved, it is perturbed by removing complete routes and reconstructing new ones. One distinctive decision is that, for constructing a route, they consider having at most two subtours.

Lum et al. (2015) follow a different approach. The SB-VRP instance is transformed into a classical VRP, where an initial solution is obtained by developing a simulated annealing (SA) using a standard algorithm for the VRP provided in `VRPH` (Groer 2012). Then, since this solution may be of poor quality, a post-processing stage and a variable neighborhood descent (VND) exploiting five specifically designed local search operators are considered to obtain the final solution. The results are good in general, also evaluated on the instances provided up to the preselection stage, where 10 of the overall 27 teams classified for the final round.

Absi et al. (2015) propose a *relax-and-repair* approach. In the first phase, a relaxation of the SB-VRP is considered by discarding accessibility constraints and solving an heterogeneous-fleet VRP (H-VRP), where the different types of trucks cover the feasible configurations allowed by the SB-VRP. For this phase, a parallel memetic algorithm is considered. Since the outcome can be an infeasible SB-VRP solution, a repair procedure is considered to try to recover from infeasibility incurred by customers that can be visited only by trucks carrying only one swap body. Finally, a post-optimization procedure is executed where all feasible routes identified along the first two phases are considered in a set-partitioning problem in order to select the best ones covering all customers.

The purpose of this paper is to contribute to the development of algorithms and approaches for the SB-VRP by describing the details of our approach in the context of the competition, where a real-world context is simulated by providing a particular running environment and limited computation time to solve real-world instances. We first point out some basic properties regarding the SB-VRP and then describe our approach, that ranked in the second position of the final round of the VeRoLog Solver Challenge 2014 (Heid et al. 2014). Two different clustering techniques are explored and evaluated computationally, obtaining as a result an improved version of the algorithm. We also make an explicit comparison with the results obtained by Huber and Geiger (2014), Lum et al. (2015) and Absi et al. (2015), and analyze the results obtained to provide further details of the behavior of our approach.

The rest of the paper is organized as follows. In Sect. 2 we include a literature review of problems related to the SB-VRP and describe some observations and remarks regarding the characteristics of the problem. Then, in Sect. 3 we describe our metaheuristic approach. Finally, in Sect. 4 we present the computational results over the benchmark instances used in the competition and in Sect. 5 we draw some conclusions and propose future research directions.

2 The SB-VRP

2.1 Problem definition

The SB-VRP is defined in Heid et al. (2014). We follow most of the notation introduced by Huber and Geiger (2014) for the definition of the SB-VRP. The main distinctive feature of the SB-VRP with respect to other well known problems from the VRP literature is the characteristic of the fleet. An unlimited homogeneous fleet of vehicles consisting of *trucks*, *semi-trailers* and *swap-bodies* start the operations at the depot v_0. Two types of vehicles are allowed to start at the depot, which are a *single truck*, carrying one swap body, or a *train* composed by a truck, a semi-trailer and two swap-bodies. Swap-bodies are assumed to be identical, each of them with a fixed capacity. Only swap-bodies can be loaded to a truck.

Consider a complete digraph $G = (V, A)$, where V is the set of vertices and A the set of arcs. Three different types of vertices are considered: vertex v_0 represents the central depot, a set of *swap locations* $V_s = \{sw_1, \ldots, sw_m\}$, where vehicles are allowed to perform certain actions, and a set of customers $V_c = \{v_1, \ldots, v_n\}$, where $V = \{v_0\} \cup V_s \cup V_c$. Each arc $(i, j) \in E$ has associated a non-negative travel distance d_{ij} and a non-negative travel time t_{ij}. Customer $v_i \in V_c$ has an associated demand $q_i \geq 0$, which must be served by exactly one vehicle, and an associated property indicating whether it can be visited only by a truck or by a train as well. The latter accounts for certain customers located in areas where, due to shunting constraints, vehicles carrying more than one swap body cannot access. The former are usually called *truck customers*, while the latter are called *train customers*.

The swap locations can be optionally used by the vehicles—and they are the only places where it is allowed—to perform certain actions with the swap bodies carried by a train. All swap locations allow the same actions to be performed. Furthermore, there is no limit on the number of times a vehicle can visit a swap location and no synchronization is required if two or more vehicles intend to use the same swap location. The valid actions allowed are four: *park*, *pick-up*, *swap* and *exchange*, where t_{park}, t_{pick}, t_{swap} and t_{ex} represent the time required for each action, respectively. These actions modify temporarily the original setting of a train by detaching, picking up and swapping a swap body carried by a train. Figure 1 describes

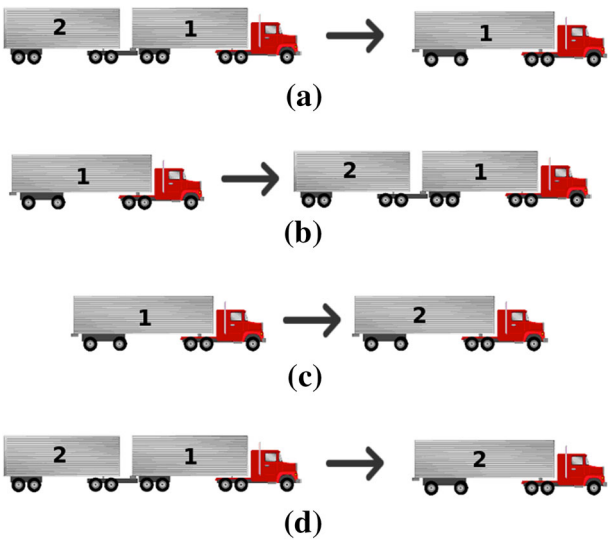


Fig. 1 Swap actions. **a** Park, **b** pickup, **c** swap and **d** exchange

the behavior of each action. As a consequence, this allows a vehicle that started a route as a train to adapt its structure and visit a truck customer. We name to these particular portions of a route a *subtour*.

Only swap bodies can be loaded, each of them with a capacity Q. Therefore a single truck has a capacity Q while a train has capacity $2Q$. Transfers of swap bodies within a route are not allowed, and the vehicle must return to the depot with the same swap bodies as originally departed.

Regarding operational constraints, the demand for each customer must be satisfied by exactly one visit of a vehicle, taking into account train limitations for truck customers. Each route must not exceed the capacity of the vehicle. In this sense, it is important to remark that it is not allowed to transfer load from one swap body to another one within a route and that swap bodies cannot be exchanged among vehicles at swap locations. In addition, the route duration, including the time spent at swap locations performing actions, must not exceed a fixed maximum duration limit T.

The structure of the objective function includes several parameters, which allow the company to obtain a reasonable approximation of the real costs incurred during operations. There are fixed costs f^{t} and f^{s} for the use of a truck and a semi-trailer, respectively, and there are no costs related to the use of swap bodies. Variable costs $ct_{\mathrm{d}}^{\mathrm{tr}}$ and $ct_{\mathrm{t}}^{\mathrm{tr}}$ are considered for the distance traveled and the time spent, respectively, for a truck. In addition, for trains, a variable cost $cs_{\mathrm{d}}^{\mathrm{semi}}$ is considered for the total distance the semi-trailer is used. Note that, in general, actions at swap locations can be used to visit a truck customer within a train route but, also, to eventually reduce the cost of a route and of the overall solution by avoiding the costs incurred when carrying a semi-trailer.

The SB-VRP consists in finding a set of feasible routes, covering each customer exactly once and satisfying the operational constraints mentioned before, at minimum total cost. To illustrate the characteristics of the problem, we provide in Fig. 2 a toy example with the basic structure of a feasible solution for the SB-VRP. In this example we can observe a pure truck route, a train route, and a train route with two subtours. For the latter, next to each portion of

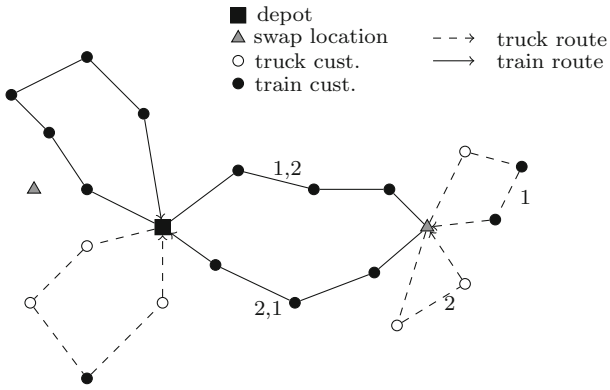


Fig. 2 Example of a feasible solution

the route it is indicated which swap bodies are carried. For this particular solution, the first visit to the swap location is a park, the second one is a swap, and finally a pickup. Observe also that it is not mandatory to visit all swap locations.

Finally, we make a practical remark on the information of the instances. As specified in the problem definition in Heid et al. (2014), both the distance and travel times are the result of a combination of the shortest path calculation including the distance and the driving times and do not necessarily satisfy the triangle inequality. In addition, they are both considered as asymmetric. However, input files provide for each location (depot, customer, and swap location) the corresponding coordinates. We observe that they keep a relation with both the distance and driving times. Our approach explicitly uses this information at the beginning of the algorithm and, therefore, we assume this extra information to be part of the instance.

2.2 Literature review and relation with other problems

A similar problem to the SB-VRP is the truck and trailer routing problem (TTRP) which is, to the best of our knowledge, introduced by Semet and Taillard (1993) and later considered in Semet (1995). The TTRP shares some of the characteristics of the SB-VRP but, however, presents the following differences:

- it considers a limited fleet, and trucks and trailers may have different capacities (i.e., carrying swap bodies with different capacities);
- actions can be performed at train customers, i.e., there are no swap locations;
- the actions allowed are park, pickup, and shifting demands loads, indicating that the semi-trailer is leaved at a feasible location and the truck travels alone;
- there is no time limit on the duration of a route.

In the last few years, the TTRP has caught some attention in the scientific community and several metaheuristic approaches for the problem are proposed in the related literature. Chao (2002) proposes a tabu search method which is applied to 21 instances having between 50 and 200 customers, with different configurations. Scheuerer (2006) builds upon this paper and proposes two new construction heuristics, which are evaluated in a similar framework as in Chao (2002). The results obtained over the 21 benchmark instances showed the approach to be effective, obtaining better solutions in all cases. Lin et al. (2009) consider a Simulated Annealing heuristic that improved the best solution found by Scheuerer (2006) in 17 of the 21 instances, and showing mixed results regarding the average results. These results are slightly improved in a follow-up paper in Lin et al. (2010). Caramia and Guerriero

(2010) propose a two-stage *Matheuristic*, assigning customers to routes at first, and then optimizing each route locally. Villegas et al. (2011) develop a greedy randomized adaptative search procedure (GRASP) with a path-relinking strategy for the problem, outperforming the previous approaches and reducing the variability in the results. Finally, Derigs et al. (2013) propose a framework for the TTRP that can be adapted to manage other variants, i.e. time windows and the case without load transfer. They make an important observation regarding the instances, claiming that some of them as well as the definition of the problem overlook several important aspects.

Gerdessen (1996) considered the vehicle routing problem with trailers (VRPT), having a similar structure as the TTRP and with applications in the distribution of dairy products in rural areas. In this problem there are no accessibility constraints but, however, the difference relies in the service time required to serve a customer.

Regarding generalizations of the TTRP, Tan et al. (2006) consider the TTVRP, a multi-objective optimization problem where trucks can be used to pickup and deliver goods and are required to visit exchange points in order to select the correct trailer type. They propose an ILP formulation and the problem is tackled using an evolutionary algorithm. Computational results are shown for instances having at most 132 jobs (customers) considering two different objectives: route cost and the number of trucks used. Drexl (2013) studies the VRP with trailers and transhipments (VRPTT), a very general problem which captures several synchronization aspects regarding time, load and spatial constraints. A formal framework including the definition of a particular network for such problems is discussed. Recently, Drexl (2011) studies the generalized TTRP(GTTRP), in which swap locations (called *transhipment locations*) are considered. Vehicles are also allowed to transfer their loads and trailers can be dropped and picked up by another compatible truck.

2.3 Properties and remarks for the SB-VRP

We begin by pointing out a few observations regarding the SB-VRP, mainly focused on limits and characteristics imposed by the operational constraints, which are later incorporated in our algorithm.

The first observation concerns the use of swap locations, both in terms of the feasibility as well as the cost of a solution. Consider the two routes in Fig. 3, visiting the same set of locations. In Fig. 3a, the customers visited next to the swap location are close, while in Fig. 3b the same swap location is next to more remote customers. The presence of a maximum travel time duration T for a route clearly limits the possibility of visiting a swap location to perform a sequence of activities. In addition, even when the maximum duration of a route is not exceeded, obtaining a significant decrease in the cost of the route will depend on the relation between the values $ct_{\mathrm{d}}^{\mathrm{tr}}$ and $cs_{\mathrm{d}}^{\mathrm{semi}}$. Assuming that the distance and travel time keeps

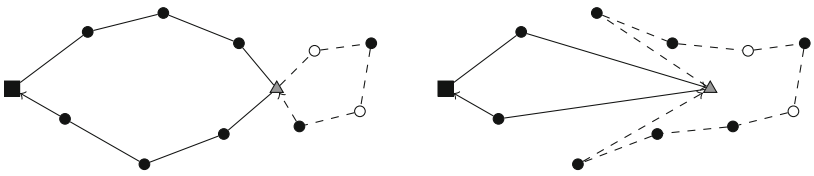


Fig. 3 Examples of routes. **a** Swap location next to close customers, **b** swap location next to far customer

a relation with respect to their relative location, the route shown in Fig. 3b is larger (both in time and distance) compared with the one in Fig. 3a.

This simple analysis shows that for some cases, limiting the use of a swap location could represent a good alternative to reduce the size of certain local search neighborhoods, if the customers to be visited immediately after/before it are close enough, without losing many feasible solutions. Indeed, as explained later, we consider for each customer only a restricted list containing a predefined subset of the (closest) swap locations.

We also analyze how capacities are handled in a train. Operational constraints establish that transfers of load between swap bodies are not allowed, even when they belong to the same train. However, as stated in the definition, it is valid to load the quantity for one customer on the two swap bodies—and it must be visited by the train—and the order of the swap bodies is not relevant. We therefore consider the following result.

Remark 1 Let $r = \langle v_0, i_1, \ldots, i_k, v_0 \rangle$ be a feasible sequence of vertices visited by a train. Define $\Gamma^r \subseteq V_c$ the set of customers visited while carrying the two swap bodies and $\Gamma^r_l \subseteq V_c$ the subset of customers visited while carrying only swap body $l, l = 1, 2$. Then, r is feasible in terms of capacity iff

$$\sum_{v \in \Gamma^r_l} q_v \leq Q, \quad l = 1, 2 \tag{1}$$

$$\sum_{v \in \Gamma^r} q_v + \sum_{v \in \Gamma^r_1 \cup \Gamma^r_2} q_v \leq 2Q. \tag{2}$$

These two conditions can be easily translated into an straightforward algorithm that in $O(|r|)$ can check the feasibility of a route in terms of the capacity constraints. Therefore, capacity can be handled more generally in terms of feasibility and previous decisions regarding which swap body serves to each customer do not restrict traditional operations when altering the structure, order and/or composition of a route.

Finally, we make an observation regarding the first operation at a swap location of a route. Based on the definition of each operation, it is reasonable (and indeed happens in all test instances provided) that $t_{\text{park}} \leq t_{\text{ex}}$, given that an exchange can be seen as a park followed by a swap. Under this scenario, the following result holds.

Remark 2 If $t_{\text{park}} \leq t_{\text{ex}}$, then the first action at the first swap location visited by a route, if any, should be a park.

The rationale behind this remark is very intuitive. The main difference between park and exchange operations is which swap body is selected for a simple subtour. Since we are considering the first action at the first swap location and customers allocated before in the route are visited by the train with the two swap bodies, it is possible to simply invert the role of each swap body in the rest of the tour and use the result in Remark 1 to allocate the corresponding capacities. If $t_{\text{park}} \leq t_{\text{ex}}$, the overall time for the route is reduced, without affecting the feasibility, and the cost of the route is improved as well.

3 Proposed approach

In this section we provide the details of the approach developed for the competition, including two basic greedy algorithms which are quite intuitive and were first considered as initial approaches as well as to provide a baseline for our evaluation. In addition, an alternative

clustering method is considered, which is later evaluated and compared with the original approach in Sect. 4. It is important to remark that some of the decisions taken regarding the structure of the algorithm are driven by the context imposed by the competition, where limited resources and time are available to solve real-world instances.

3.1 General comments

We now proceed to describe a few decisions made during the development of the solution method for the competition, which somehow affect the different parts of the approach. The first one involves the feasibility check in terms of capacity. Following the result in Remark 1, we consider a straightforward algorithm to determine whether a particular operation (i.e., insertion, exchange, etc.) is feasible or not in terms of the total capacity. This allows us to avoid handling explicitly which swap bodies satisfy the demand of each customer, providing as well a more general framework in this regard.

We also take into account the observation made in Sect. 2.3 regarding the proximity among customers and swap locations. Therefore, we define for each customer $v_i \in V_c$ a restricted set of swap locations, $\hat{Sl}(v_i)$, to capture this effect. Given a swap location $sw_j \in V_s$, the following (heuristic) *weighted cost* incurred when traveling arcs (sw_j, v_i) and (v_i, sw_j) using only a truck is

$$\hat{c}_{v_i sw_j} = \hat{c}_{sw_j v_i} = \alpha \left(ct_{\mathrm{d}}^{\mathrm{tr}} \times \left(d_{v_i sw_j} + d_{v_j sw_i} \right) \right) + (1 - \alpha) \left(ct_{\mathrm{t}}^{\mathrm{tr}} \times \left(t_{v_i sw_j} + t_{v_j sw_i} \right) \right) \quad (3)$$

Then, we define $\hat{Sl}(v_i)$ as the K *closest* swap locations to v_i according to the weighted cost $\hat{c}$. The motivation behind this definition is to capture swap locations which are likely to be feasible and also incurring in a small cost when accessing to them. Indeed, we observed on preliminary computational experiments that considering these sets instead of all the swap locations reduces significantly the computation times of certain routines, while obtaining similar results.

3.2 Constructive heuristics

These two greedy algorithms could represent a very first approach for the SB-VRP, and were used for comparison purposes. In addition, a straightforward extension of the greedy approach presented in Sect. 3.2.2 is used as the constructive step in a GRASP metaheuristic, as described later in the document.

3.2.1 Greedy naive

The first heuristic we consider is a simple heuristic which iteratively inserts unassigned vertices into the partial solution following a greedy criterion, while in case no feasible insertion can be made it forces the opening of a new route. This heuristic does not account for swap locations, but it is used to provide a baseline for comparison purposes. In Algorithm 1 we show the sketch of the greedy naive (GN) heuristic.

We note the two cases described in Step 2, when opening a new route. Firstly, when considering a truck customer the only alternative is to open a simple route since swap locations are not considered. For a train customer, however, both types of routes can be considered. Due to the greedy selection criterion based on the cost incurred, only truck routes would be opened. Therefore, to reduce the final number of routes we decided to open single routes only when necessary. On preliminary computational results, this particular distinction produced better results, incurring in overall smaller fixed and variable costs.

Algorithm 1 Greedy naive (GN) heuristic

Input: SB-VRP instance.
Output: Feasible solution.

1. (Initialization) Let $S = V_c$ the set of unassigned customers. Let $R = \emptyset$ be the set of routes representing the (partial) solution.
2. (Evaluation) For each $v \in S$, compute the cost $\hat{c}_v^{ir}$ incurred by inserting v in the position i of route $r \in R$, if feasible. Consider also the cost $\hat{c}_v^{0s}$ incurred by opening a new single route to visit v if it is a truck customer, and the cost $\hat{c}_v^{0t}$ of a new complete route to visit v if it is a train customer.
3. (Insertion) Select the vertex v^* producing the least cost insertion $\hat{c}_{v^*}^{ir}$ for position i in route r. If r is a new route, set $R = R \cup \{r\}$ (where r is a single route or a train, depending on the choice). Insert v^* in position i of route $r \in R$. Set $S = S \backslash \{v^*\}$.
4. (Termination) If $S \neq \emptyset$, return to Step 2. Otherwise, return the solution and terminate the algorithm.

3.2.2 Greedy with swap locations

We now extend the idea described in the previous section to include actions and operations at the swap locations. For this purpose, when attempting to insert a vertex into an existing route, we will also try to begin a subtour or, if already within a subtour, try to perform a swap action. In all cases, the feasibility of the solution in terms of capacity, duration and structure of the swap activities is maintained.

We first show in Algorithm 2 a sketch of the insertions attempted for an unassigned vertex into a particular position within an existing route in the current solution. Recall the notation introduced in the previous section, where S denotes the set of unassigned vertices and R the set of routes in the partial solution.

Algorithm 2 Vertex insertion evaluation

Input: SB-VRP instance, an unassigned vertex $v \in S$, route $r \in R$, position $i = (v_a, v_b) \in r$
Output: Cost of the best insertion of vertex v, if any.

1. Evaluate the cost of the following insertions, if feasible:
   - insert v alone between v_a and v_b;
   - if r is a train route and i is within a subtour, let sw be the active swap location. Evaluate inserting the sequences $\langle sw, v\rangle$ and $\langle v, sw\rangle$;
   - if r is a train route and when traversing i the vehicle carries both swap bodies, evaluate inserting a *single subtour* composed by the sequence $\langle sw, v, sw\rangle$, for $sw \in V_s \cap \hat{Sl}(v)$.
2. If at least one of the previous attempts is feasible, return the cost of the best one.

In Algorithm 3 we show the details of a greedy heuristic that incorporates the use of swap locations, which we name G-SL. The main idea behind the algorithm is similar to GN, except for two minor differences. When attempting to insert a vertex into an existing route we call Algorithm 2 to consider the use of swap locations. Secondly, contrary to GN, new routes are opened only when no feasible insertion in the actual routes is possible, despite eventually opening a new route could incur in a smaller cost. This is expected to increase the chances of using swap locations in the solution.

Algorithm 3 Greedy with swap locations (G-SL) heuristic

Input: SB-VRP instance.
Output: Feasible solution.

1. (Initialization) Let $S = V_c$ be the set of unassigned customers. Let $R = \emptyset$ be the set of routes representing the (partial) solution.
2. (Vertex insertion) For each unassigned vertex $v \in S$, route $r \in R$, and position $i = (v_a, v_b)$ evaluate the insertions as described in Algorithm 2. If at least one feasible insertion can be done, select and perform the best one. Let v^* be the vertex involved, update $S = S \backslash \{v^*\}$ and go to Step 4.
3. (Route extension) If no feasible insertion has been found in the previous step, attempt an insertion in a new route. When considering a train route, attempt insertion using Algorithm 2 as in the previous step. Select and perform the best insertion. Let v^*, r^* be the selected vertex and the new route, respectively. Set $S = S \backslash \{v^*\}$ and $R = R \cup \{r^*\}$.
4. (Termination) If $S \neq \emptyset$, return to Step 2. Otherwise, return the solution and terminate the algorithm.

3.3 Local search operators

In this section we provide some details regarding the local search operators considered, the way in which they are adapted to the SB-VRP and which is the neighborhood we consider. The possible movements are the following:

1. *Exchange* For every two customers $v_i, v_j \in V_c$ allocated in different routes, exchange their positions.
2. *Relocate* For every customer $v_i \in V_c$, remove it from its current position and attempt to insert it in every other position of the current solution.
3. *Restricted Or-Opt* Within a route, each sequence of consecutive customers having size $l = 1, 2, 3$ is removed and several insertions are attempted in the remaining positions of the route. Let s be the sequence of customers, with v_i^s and v_f^s the beginning and the end of the sequence, respectively. Let also $r^* \in R$ be the route obtained by removing s and let $i \in r^*$ be an arc. Then, we attempt the following insertions at i:
   - insert s and its reverse, noted $rev(s)$,
   - if i belongs to a subtour, with $sw \in V_s$ the active swap location, test the sequences $\langle sw, s\rangle$, $\langle s, sw\rangle$, $\langle sw, rev(s)\rangle$ and $\langle rev(s), sw\rangle$,
   - if i belongs to a train sector of the route, for each $sw \in \hat{Sl}(v_i^s) \cap \hat{Sl}(v_f^s)$ attempt to insert the complete subtours $\langle sw, s, sw\rangle$ and $\langle sw, rev(s), sw\rangle$ by performing a park.
4. *Restricted 2-Opt* For the SB-VRP, a train route using one or more swap locations could easily fall into an infeasible solution. Since the 2-Opt operator needs to revert a subpath in the route, if a swap location is contained in such subpath it is necessary, at least, to define how this is handled. Furthermore, truck customers visited within a subtour could be easily moved outside to a train section of the route, violating one of the operational constraints. Therefore, we decided to apply the 2-Opt operator in an intra-route fashion, and within subpaths where the composition of the truck does not change. That is: within each subtour and on subpaths conformed by consecutive vertices visited by a train, if any. To clarify, in case no swap locations are present in a route, the operator behaves as in its standard definition. In addition, this adaptation allows us to obtain a faster operator since some moves are omitted.
5. *Route downgrade* For each train route such that the total demand of all visited customers does not exceed Q, remove the swap locations (if any) and create a new simple truck route that visits the customers in the same order and carrying only one swap body.

These operators are applied in the order they are presented, in a *best improvement* fashion, and our implementation moves from one operator to the next one when no improvement can be found. This sequence is applied iteratively until no better solution can be achieved by any of the operators.

We make a brief discussion regarding some of the operators presented below. Firstly, we remark that the Or-Opt we consider, although restricted to a route, includes the evaluation as a possible movement not only the reverse of a sequence but, in addition, the insertion within a subtour or even the possibility of starting a new one within a train route. The motivation for this decision is to allow new subtours to be created, both aiming to visit truck-only customers and also to eventually reduce the overall cost of the tour. With respect to the downgrade route, we experimentally observed that this is a common situation after exploring the neighborhood. To facilitate the use of swap locations in a train route, we execute it only at the end of the process.

3.4 Shaking procedure

To escape from local optimum solutions, one of the alternatives considered is to apply a shaking procedure to the solution and restart the search from the new generated solution. For this purpose, given a feasible solution for the SB-VRP, each customer is removed from the solution with uniform probability $p \in (0, 1)$. Let L be the list of removed customers. To reconstruct a feasible solution we shuffle L and its vertices are sequentially considered and inserted into the restricted solution in the position and route incurring in the smallest cost increment.

3.5 GRASP

Another alternative considered to escape from a local optimum is to randomize the construction phase, as proposed by the well known GRASP (see, e.g., Feo and Resende 1995). For each initial solution, a local search procedure is executed in order to improve the solution. Therefore, we adapt and include into this scheme the procedures described before.

For the construction phase, we develop a straightforward adaptation of Algorithm 3, where instead of returning the best feasible insertion we construct a *restricted candidate list* having the best $rclsize$ insertions. To decide which insertion is performed, we randomly select one of the elements in the list. We further add another random decision to the heuristic to decide which type of route to open. Unless all the remaining unassigned customers are required to be visited by a train (i.e., $Q < q_v \leq 2Q$, for all unassigned $v \in V_c$), a single truck route is open with uniform probability p_r and a train route with probability $1 - p_r$, $p_r \in (0, 1)$. We consider generating a maximum of nit_{const} initial solutions.

Regarding the improvement phase, we consider an ILS combining the neighborhood defined in Sect. 3.3 and the shake procedure from Sect. 3.4. Starting from a feasible solution, we explore the neighborhood until no improvement can be found, apply the shaking procedure and repeat the process. The shaking procedure is applied always using the best solution obtained from the last constructed solution until the limit of nit_{ils} consecutive iterations without improvements is reached. After reaching this limit, if possible, we start over again with the constructive phase. The sketch for the overall procedure is shown in Algorithm 4.

3.6 Cluster-first route-second: competition approach

We point out the last decision concerning our strategy for the challenge, where the evaluation is performed considering large size instances with approximately 500 customers and 150 swap

Algorithm 4 GRASP heuristic for the SB-VRP

Input: SB-VRP instance, nit_{const}, nit_{ils}, f objective function
Output: Feasible solution.

1. (Initialization) Set $it = 0$ and $z_{\text{best}} = \infty$.
2. (Construction phase) If $it \geq nit_{\text{const}}$, go to Step 5. Otherwise, construct a feasible solution using the greedy randomized heuristic. Set x_{cur} as the solution. Define $it_{\text{ils}} = 0$ and $it = it + 1$.
3. (ILS) Explore the neighborhood defined in Sect. 3.3. Let $\hat{x}$ be the solution obtained. If $f(\hat{x}) \leq f(x_{\text{best}})$, update $x_{\text{best}} = \hat{x}$.
4. (Evaluation) If $f(\hat{x}) \leq f(x_{\text{cur}})$, $x_{\text{cur}} = \hat{x}$, $it_{\text{ils}} = 0$ and return to Step 3. If $it_{\text{ils}} > nit_{\text{ils}}$, go to Step 2. Otherwise, shake solution x_{cur}, set $it_{\text{ils}} = it_{\text{ils}} + 1$ and return to Step 3.
5. (Termination) Return solution x_{best} and $f(x_{\text{best}})$.

locations. For this type of instance, the execution time of both the constructive algorithms as well as the local search operators can be considerably increased. To tackle this issue, for example, Lum et al. (2015) restrict the set of neighbors of each vertex during the search. We adopted a similar strategy regarding the swap locations. However, for the remaining we decided to follow a *cluster-first route-second* approach, which takes into account the upper limit on the running time for the program established by the organizers.

Given the complete instance, we divide the vertices (customers and swap locations) into at most ncl clusters using *k-means* taking as input the coordinates of the vertices, for some value ncl set a priori. Once each cluster has been determined, they are treated independently for a certain amount of time and, at the end, the best solutions found for each cluster form a complete and feasible solution for the instance.

The procedure is as follows. Let $T^{\max}$ be the time limit for running the algorithm and ncl the number of clusters. Since our program for the competition runs in a single thread and executes sequentially, we assign at most 2/3 of $T^{\max}$ to optimize the clusters independently, where this time is distributed among the cluster proportionally with respect to the number of customers. Each optimization is carried out using the GRASP procedure described in Sect. 3.5. When each cluster has either finished the procedure or reached the upper limit established, we dedicate the remaining 1/3 of the time to optimize the merged solution by executing the ILS procedure described in the previous section. When the overall time limit is reached, the best solution found during the process is reported.

Based on limited preliminary experiments, we established the following parameters for the algorithm. Firstly, we observed that dividing the instance in at most four clusters provided good results. For the CFRS-vsc, we set $nit_{\text{const}} = 50$ and $nit_{\text{ils}} = 150$. This configuration aims to provide the following behavior: for small and medium instances, when exploring the search space starting from the best solution stops finding improvements, then we start over by recomputing an initial solution. However, for large instances, exploring the neighborhoods is time consuming and we prefer to restrict our attention to the ILS. Indeed, with the time limits imposed by the competition, this is the standard behavior of the GRASP procedure. For the weighted cost, we set $\alpha = 0.5$. Within the greedy randomized, the probability of opening a single truck route is set to $p_r = 0.2$, $rclsize = 5$, we consider the $K = 10$ closest customers and $p = 0.15$ as the probability of removing a customer in the shaking procedure if the instance (or cluster) has $n \geq 200$, $p = 0.3$ if $50 \leq n < 200$, and $p = 0.5$ otherwise. For the rest of the paper, we name CFRS-vsc to the method described in this section.

3.7 Alternative clustering and general scheme

Even when the *k-means* approach represents a significant improvement, in particular for large instances, we observed that in some cases it may produce a smaller number of clusters than expected (due to a random centroid initialization) since some clusters *disappear*. In addition, the partition is not necessarily balanced with respect to the size of the clusters. Although this is somehow considered when we proportionally assign the time to each cluster with respect to the number of customers, we further include an alternative clustering method.

The method is the so-called *angular clustering*, where customers (and swap locations as well) are scanned and added to a cluster according to their angle with respect to the depot. The procedure is similar to the one described in Gillett and Miller (1974). The coordinates provided for each vertex in the input files are taken as the ones to compute the angle. For each vertex we compute the angle respect to the depot, which are then sorted increasingly. Finally, k clusters are constructed aiming to have approximately n/k customers, which results in a balanced partition.

Regarding the general approach, we consider different configurations for some of the parameters aiming to evaluate the impact of the clustering phase. Firstly, the parameter k and its impact is evaluated experimentally in the next section. In addition, we consider an implementation where each cluster is tackled separately in an independent thread for $T^{\max}$, instead of using a proportion of the time for each cluster. Finally, we considered different settings for $T^{\max}$ to asses the impact of this parameter within the approach. With this experiments we can provide an insight regarding the cluster-first route-second approach, which is complimentary to the approaches and investigations proposed for the SB-VRP.

4 Computational experiments

The methods described in the previous section have been implemented in C++, using `g++ 4.8.4` and an Ubuntu Linux 14.04 LTS as operating system. The experiments are run on a Workstation with an Intel Core i7-2600 3.4 GHz CPU and 16 Gb of RAM. We remark that this configuration is slightly different from the one used in the competition (i.e., the operating system used for the competition was Windows). Although it was possible to use four processors, the version of our program submitted to the competition runs in a single thread. The multi-thread version of the methods has been implemented using *OpenMP* (Open MP Architecture Review Board 2011). The time limit $T^{\max}$ used both in the competition and as well in our experiments is 600 s. The methods use always all the time available, although in some cases, particularly in small instances, the best solution is found early in the execution.

Regarding the instances, we consider the ones provided by the organizers during the competition. Three different sets of instances were initially made public: a *small* set, having approximately 50 customers and 20 swap locations; a *medium* instance, with nearly 200 customers and 40 swap locations; and a *large* instance, having more than 500 customers and 100 swap locations. All instances have three different versions: a *normal* version, where the type of customers is mixed; an *all with trailer* version, where all customers can be visited by a train; and an *all without trailer*, where customers can only be visited by a single truck. To identify each type of instance, we add a suffix *n* for *normal*, *awt* for *all with trailer* and *awot* for *all without trailer*.

A first *preselection round* took place at the middle of the competition, where the best ten teams classified for the final round. For this preselection, instances *medium normal* and *large normal* where considered, as well as a new large instance with its two corresponding

variants regarding the type of customers considered, giving a total of five instances. The final round was similar, where two new large instances and their corresponding variants were considered, giving a total of six instances. We refer the reader to Huber and Geiger (2014) for more details regarding the instances up to the preselection.

The rest of this section is divided in two parts. Firstly, we report a summary of the experiments conducted to evaluate some of the different parameters of the algorithm. From the preliminary experiments we observed that one of the key components of the approach is the clustering phase, mainly for large instances. Thus, experiments evaluating different configurations are reported, aiming to provide an insight on the overall approach. Huber and Geiger (2014) provide an extensive analysis for similar neighborhood operators as the ones considered in this paper. Absi et al. (2015), among other results, conduct a set of experiments in order to asses the impact of an implementation exploiting a multi-core machine. We also report results in this direction and its impact when combined with the clustering approach.

Secondly, we compare the best configuration identified for our approach with the other methods from the literature on all the instances available from the competition. In addition, for the final instances, we also compare the results obtained with the greedy approaches described in the previous section in order to provide a baseline.

Regarding the methods, for this section we consider the following:

- CFRS-vsc: method described in Sect. 3.6, which represents the submission to the competition. Unless otherwise stated, we assume $k = 4$ and $T^{\max} = 2/3$ of the overall execution time available, running on single thread.
- CFRS-ang: variant described in Sect. 3.7, where k stands for the number of clusters considered.

For the results, we report average (Avg.) and best (Best) solutions found, as well as their corresponding percentage gaps (%aG and %bG, respectively). Percentage gaps are computed as $100 \times (OBJ - BKS)/BKS$, where OBJ stands for the value of the current solution and BKS for the best known solution for the instance, which is based on results from the literature. For the latter, we refer the reader to Absi et al. (2015).

4.1 Preliminary evaluation and parameters setting

For the preliminary experiments we considered a reduced set of instances, namely *large* and *preselection* instances, and then the selected configuration is considered in the next section for the complete set of instances. For each instance and combination of parameters the average and best results over five runs are computed.

We begin by analyzing the impact of the number of clusters k on the approach submitted to the competition, CFRS-vsc, in its single-thread version. Figure 4 shows the results with the average %gap obtained by CFRS-vsc for $k = 2, \ldots, 12$, for *large* and *preselection* instances. The main message of the figure is that there is no clear tendency in all cases, specially for the *preselection* instances. Indeed, this is related to the fact that the number of clusters is at most k. However, we can observe that $k = 4$ provides good results in general. We remark that for the original selection, only values of $k \leq 6$ had been considered. A similar experiment has been conducted for the multi-thread version of CFRS-vsc, which is reported later in this section.

The same experiment has been initially conducted for the CFRS-ang, both in its single and multi-thread version. The results showed in general to be slightly better to the ones obtained with *k-means* clustering. Therefore, we decided to restrict the values of k to the range which produced the best results, $4 \leq k \leq 8$, and include $T^{\max}$ as a parameter in the

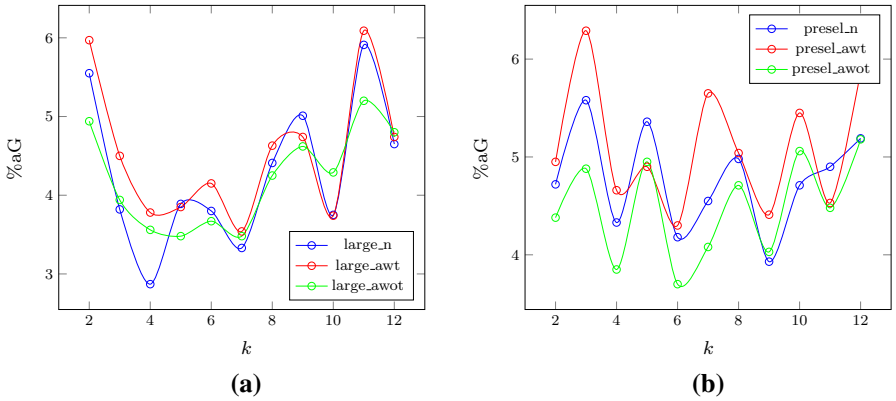


Fig. 4 Avg. %gaps for CFRS-vsc with different number of clusters. **a** kmeans, large, **b** kmeans, preselection

Table 1 Average %aG on large and preselection instances for CFRS-ang for different combinations of k and $T^{\max}$

$T^{\max}/k$	4	5	6	7	8
0.67	2.77	2.82	2.76	2.74	2.56
0.70	2.76	2.83	2.78	2.62	2.45
0.80	2.75	2.95	2.76	2.62	2.44
0.90	2.52	2.83	2.72	2.64	2.39
0.95	2.47	2.75	2.77	2.66	2.44

experiments for the multi-thread version. We consider as possible combinations $T^{\max} = 0.67, 0.7, 0.8, 0.9, 0.95$, where 0.67 represents the original setting of 2/3 of the available time. Table 1 shows the average %gap over the six instances for each combination of k and $T^{\max}$.

The results over all instances show small differences among the combinations, although we observe a little consistent advantage for $k = 8$, where the best combination is with $T^{\max} = 0.9$. It is interesting to observe that for $k = 4, 5$ the gaps tend to decrease when $T^{\max}$ increases, and the best results are achieved with 0.95. However, for $k = 6, 7, 8$ results for $T^{\max} = 0.95$ are slightly worse than for 0.9. This can be explained by the fact that solving each cluster separately is clearly suboptimal and, when the number of clusters is larger, the operators are able to find improvements to the initial solution.

The aggregated results from Table 2 cover some interesting behavior of the approach, related to the fact that the best number of clusters is highly coupled with the instance. In Table 2 we report the %aG for each instance and value of k, averaged over the values of $T^{\max}$. The main observation from this table is that the number of clusters has an impact on the quality of the solutions obtained. For instance, $k = 8$ obtains the best solutions on the *large* instances, improving by 1.4 % on average the results obtained by $k = 5$ in two of the three instances. On the contrary, $k = 5$ obtains the best average results for *preselection* instances, improving by nearly 1 % the results produced with $k = 8$ on the last instance. Overall, although all percentages are within a small range, our selection for CFRS-ang is setting $k = 8$ and $T^{\max} = 0.9$.

We next show the results regarding the impact of using a multi-thread version (running on four threads) of the algorithm. Table 3 shows the %aG for the single and multi-thread

Table 2 Avg. %aG for each instance and k, averaged over $T^{\max}$

Instance	k				
	4	5	6	7	8
large_n	2.27	3.10	2.58	2.35	1.76
large_awt	2.47	3.52	2.81	2.17	1.92
large_awot	2.26	3.07	2.16	2.40	2.27
presel_n	2.88	2.25	2.76	2.75	2.75
presel_awt	3.42	2.61	3.25	3.06	2.81
presel_awot	2.64	2.48	2.99	3.20	3.23
Avg.	2.66	2.84	2.76	2.66	2.46

Table 3 Single versus multi-thread avg.% gaps

Instance	CFRS-vsc				CFRS-ang			
	Single thread		Multi-thread		Single thread		Multi-thread	
	%aG	%bG	%aG	%bG	%aG	%bG	%aG	%bG
large_n	2.87	2.52	2.68	1.70	2.65	2.21	1.76	1.41
large_awt	3.78	3.38	3.25	2.15	2.60	1.95	1.76	1.38
large_awot	3.56	2.35	2.88	2.38	3.10	2.80	2.02	1.91
presel_n	4.33	3.48	3.00	2.26	3.76	3.42	2.77	2.33
presel_awt	4.66	3.39	3.40	2.63	3.44	2.95	2.79	2.63
presel_awot	3.85	3.71	3.09	2.42	3.74	3.35	3.23	2.93

version of the selected configurations for each algorithm, namely CFRS-vsc with $k = 4$ and $T^{\max} = 0.67$ and CFRS-ang with $k = 8$ and $T^{\max} = 0.9$. The results are somehow aligned with the ones reported by Absi et al. (2015), obtaining on average improvements of around 1%. The results seem to be more consistent for CFRS-ang, possibly due to the fact that the size of the clusters is balanced.

Finally, we comment on some particular behaviors observed during the experimentation. We noted that CFRS-ang produced, consistently, better results than CFRS-vsc, although in some instances this is not the case. In addition, we observed that the structure of the instance plays an important role regarding the number of cluster to consider, as can be noted from the results in this section.

4.2 Comparison with other methods

We begin by showing in Table 4 the results obtained by CFRS-vsc and CFRS-ang on the instances available after the preselection phase. We report for each method the average (Avg.) and best solution found (Best) over ten independent runs with different seeds. As reported in the previous section, CFRS-ang produces in general better results than CFRS-vsc on big instances (*large* and *preselection*). However, we can observe that CFRS-vsc obtains in general much better solutions than CFRS-ang on *small* and *medium* instances, both on average and the best solution obtained. This behavior is the expected one, since the clustering approach in general and CFRS-ang in particular have been devised and experimentally tuned

Table 4 Average and best solutions for CFRS-vsc and CFRS-ang on the instances up to the preselection

Instance	BKS	CFRS-vsc		CFRS-ang	
		Avg.	Best	Avg.	Best
small_n	4797.85	4830.16	4806.97	4850.81	4815.71
small_awt	4716.58	4730.63	4730.63	4732.94	4728.93
small_awot	4839.64	4900.33	4855.62	5029.74	5002.03
medium_n	7814.17	8030.83	7942.22	8129.49	8073.45
medium_awt	7749.42	7940.37	7847.30	8043.66	7945.46
medium_awot	8021.88	8235.51	8169.69	8374.02	8318.28
large_n	20,471.40	21,020.99	20,786.70	20,817.78	20,738.50
large_awt	20,215.26	20,936.56	20,734.60	20,583.50	20,516.70
large_awot	21,176.49	21,821.13	21,641.20	21,619.78	21,522.50
presel_n	25,376.41	26,304.66	26,136.70	26,017.54	25,894.40
presel_awt	24,939.83	26,002.34	25,785.00	25,629.04	25,573.20
presel_awot	25,835.85	26,738.85	26,524.50	26,637.14	26,558.40

to tackle large instances, which are the most difficult ones. Indeed, the selection of $k = 8$ and $T^{\max} = 0.9$ may not be the best choice for medium size instances, since the clusters tend to have only a few tens of vertices, and little time is dedicated to improve the integrated solution. With respect to the approach in general, the ILS procedure and the cluster stage produced the best improvements in the overall approach. The latter, in particular, allows a better usage of the available time. Although not mentioned explicitly, the clustering phase is able to improve nearly 3–4 % with respect to previous solutions on large instances.

In Table 5 we extend and update the algorithm comparison reported in Absi et al. (2015) on the same set of instances.[1] The first column indicates the name of the instance, and BKS reports the best known solution for each instance. We then report the results for GN and G-SL where in both cases we applied to the resulting solution one local search step exploring the neighborhood described in the previous section, together with the average results for the methods proposed by Huber and Geiger (2014), Lum et al. (2015), Absi et al. (2015), as well as CFRS-vsc and CFRS-ang.

The first observation regarding the four available heuristics from the competition is that the results are comparable, obtaining solutions within the 6–7 % of the best known solution. The best results in this sense are obtained by Huber and Geiger (2014), which in several cases reached that value, and in general the solutions are below the 2 % from the BKS. Absi et al. (2015) also report good overall results, with solutions ranging below 4 % from the BKS, and showing the best performance on small and medium instances. It is important to remark that the approach in Absi et al. (2015) consider 50 % more computing time (i.e., 900 s overall) than the other approaches, where 300 extra seconds are assigned to the set-partitioning based post-optimization procedure.

Next, we focus on the results obtained by GN and G-SL, and relate them with the structure of the SB-VRP. We first remark that, although both approaches are heuristic and contain a few arbitrary design decision, they are however useful to obtain a deeper insight on the structure of the problem in general and the instances in particular. At a first glance, it seems reasonable

[1] We remark that the results reported in Absi et al. (2015) for our method are based on a preliminary draft of this research.

Table 5 Algorithm avg. comparison on the instances up to the preselection

Instance	BKS	GN	G-SL	Huber and Geiger (2014)	Lum et al. (2015)	Absi et al. (2015)	CFRS-vsc	CFRS-ang
small_n	4797.85	5890.04	6726.54	4805.32	4959.00	4808.18	4830.16	4850.81
small_awt	4716.58	5386.37	5646.86	4730.92	4873.05	4724.81	4730.63	4732.94
small_awot	4839.64	6510.88	6085.46	4860.06	5356.36	5007.74	4900.33	5029.74
medium_n	7814.17	9667.97	9591.56	7885.75	8297.25	7876.66	8030.83	8129.49
medium_awt	7749.42	9009.27	9180.53	7818.87	8335.57	7827.93	7940.37	8043.66
medium_awot	8021.88	10,338.30	9883.55	8119.43	8628.37	8112.88	8235.51	8374.02
large_n	20,471.40	24,238.60	24,342.20	20,764.05	22,051.40	21,365.48	21,020.99	20,817.78
large_awt	20,215.26	23,001.70	23,644.30	20,482.65	21,317.00	21,026.51	20,936.56	20,583.50
large_awot	21,176.49	25,935.00	25,283.60	21,457.48	22,419.40	22,026.64	21,821.13	21,619.78
presel_n	25,376.41	30,208.90	32,399.20	25,617.77	26,712.40	25,988.46	26,304.66	26,017.54
presel_awt	24,939.83	29,024.40	30,731.10	25,331.66	26,658.10	25,670.90	26,002.34	25,629.04
presel_awot	25,835.85	30,567.20	31,063.50	26,166.99	26,712.40	26,253.76	26,738.85	26,637.14
Total %diff.		19.22	21.95	1.19	5.89	2.69	3.15	2.56

Table 6 Summary of %aG on large instances for all methods

Instance	GN	G-SL	Huber and Geiger (2014)	Lum et al. (2015)	Absi et al. (2015)	CFRS-vsc	CFRS-ang
large_n	18.40	18.91	1.43	7.72	4.37	2.68	1.69
large_awt	13.78	16.96	1.32	5.45	4.01	3.57	1.82
large_awot	22.47	19.39	1.33	5.87	4.01	3.04	2.09
presel_n	19.04	27.67	0.95	5.26	2.41	3.66	2.53
presel_awt	16.38	23.22	1.57	6.89	2.93	4.26	2.76
presel_awot	18.31	20.23	1.28	3.39	1.62	3.50	3.10
Avg. %aG	18.07	21.07	1.31	5.76	3.23	3.45	2.33

to expect that by incorporating swap locations in the solution, as G-SL does, it would be beneficial. However, from Table 5 we can observe that the results of G-SL compared with GN are somehow mixed, and that GN obtained better solutions in 8 of the 12 instances. This behavior is consistent with our experience during the development of the heuristic, where for this set of instances the solutions tend to avoid using swap locations. Similar observations are reported in Lum et al. (2015).

For this set of instances, CFRS-vsc is in the middle between Huber and Geiger (2014) and Lum et al. (2015), standing within 0.5 and approximately 4 % with respect to the BKS reported in Absi et al. (2015). The results obtained on the *small* instances are comparable, and CFRS-vsc produces results around 1 % better on the *large* instances. On the contrary, Absi et al. generate better solutions for *medium* and *preselection* instances. Regarding CFRS-ang, the overall results stand between Huber and Geiger (2014) and Absi et al. (2015), but it shows a pronounced difference depending on the size of the instances. Absi et al. (2015) produce better results on the *small* and *medium* instances, specially in the latter. However, for larger instances CFRS-ang shows a better overall behavior. It generates comparable results on the *preselection* instances, and the average improvement on the *large* instances is around 2 %.

This tendency can be better appreciated in Table 6, which shows the detailed %aG for all methods for each of the larger instances, as well as the average %aG. As mentioned before, Huber and Geiger (2014) obtains the best results. When restricting to this particular set of instances, the results produced by CFRS-ang are on average 1 % better than the ones from Absi et al. (2015). Indeed, considering the results reported in Table 1, we can observe that all combinations between k and $T^{\max}$ produce better results than the approach proposed by Absi et al. (2015). Furthermore, Absi et al. also report results considering 1200 s (plus 300 s for the post-optimization) for the execution time and their approach generates solutions with an average %aG of 2.46, which is still slightly above CFRS-ang.

In general, CFRS-ang obtains better results on *normal* and *all with trailer* instances compared to the gaps for the *all without trailer* ones. We also note that CFRS-vsc produces results that are close to the ones obtained by Absi et al. (2015), only 0.2 % below on average. This suggests a reasonable performance for CFRS-vsc, since it runs in a single-thread and uses 300 s less for the execution.

We now present the results obtained by CFRS-vsc and CFRS-ang in the final set of instances. The key for the table is the same as for Table 4. We note that neither Huber and Geiger (2014), Lum et al. (2015) and Absi et al. (2015) report the results for these instances. Indeed, we only count with our results for this set of instances and the ranking provided by the organizers, where CFRS-vsc ranked in the second position. Similarly to the previous

Table 7 Results obtained on the instances used in the final round

Instance	GN	G-SL	CFRS-vsc		CFRS-ang	
			Avg.	Best	Avg.	Best
final_n	43,366.50	41,990.50	37,590.97	37,087.00	36,687.50	36,305.80
final_awt	37,529.60	41,896.80	36,805.52	36,435.50	35,449.65	34,997.90
final_awot	46,093.50	44,607.50	39,572.50	38,914.00	39,033.90	38,826.20
final_r_n	151,125.00	150,372.00	137,593.80	137,273.00	135,930.70	135,509.00
final_r_awt	139,537.00	149,361.00	132,407.10	132,072.00	131,643.00	131,445.00
final_r_awot	215,155.00	164,757.00	155,498.80	154,255.00	153,000.40	152,587.00

table, the relation between the best solution found by CFRS-vsc and CFRS-ang with respect to the average over the ten runs keeps the same relation. We remark that CFRS-ang is able to improve consistently the results obtained by CFRS-vsc (Table 7).

We can observe, however, a different trend in this table when comparing the heuristics GN and G-SL. In this case, G-SL obtains better results in 4 out of the 6 instances, and it produces worse results in the *all with trailer* instances. In addition, for the last instance, a remarkable improvement of nearly 40 % is observed between both solutions. These results are consistent with the structure observed in these instances. Compared with large instances used in the previous table, we noted that the maximum route duration has been increased considerably, and also that demands for customers represent a tight constraint in general. In particular, the difference obtained in the instance *final random all without trailer* is due to the fact that, in general, at most two customers can be allocated into a single swap body. Therefore, since all customers are truck customers, GN generates a large number of truck routes. On the other hand, G-SL attempts to use train routes trying to take advantage of the use of swap locations. As a result, a smaller number of routes is generated, where each of them visits nearly twice the number of customers with respect to the solution generated by GN.

Overall, these findings lead us to believe that the SB-VRP presents a complicated structure, and that further developments from the theoretical side as well as from a computational standpoint are required to get a deeper insight on the problem.

5 Conclusions

In this paper we study the SB-VRP and consider a cluster-first route-second with a GRASP metaheuristic to solve it. This problem is relatively new in the VRP literature and, although it presents some similarities with other VRP variants, its particular structure and the results obtained by our algorithm as well as other approaches developed simultaneously showed that it deserves further attention. Taking into account the time limit for the execution, our approaches start by splitting the instance into smaller instances, optimize them during a certain amount of time, and then merge this solution and use the remaining time to obtain a better final solution. This approach showed to be quite effective in practice, obtaining comparable results in the instances provided and ranking in the second position on the final set of instances. We also propose an alternative approach that improves the results of the ones obtained by our method during the competition and provide an explicit comparison of several approaches appearing in the literature, aiming to contribute to future developments for the problem.

As future research, based on our experience while working on the problem, we believe that more attention should be devoted to study the structure of the problem, in particular regarding the convenience (or not) of using the swap locations. In addition, taking into account the other approaches, it would be interesting to investigate the use of Integer Linear Programming both to explore large neighborhoods of a solution as well as to evaluate where is the limit for solving instances to optimality. Based on the experimental study presented in this paper regarding the clustering phase, it would be interesting to consider, if possible, using an exact approach to tackle each cluster separately.

Acknowledgments This research is partially supported by Grants PICT-2010-0304, PICT-2011-0817, PICT-2013-2460 and UBACyT 20020100100666. The authors are grateful to two anonymous referees and the associate editors for their careful reading and valuable comments, which helped improving the article.

References

Absi, N., Cattaruzza, D., Feillet, D., & Housseman, S. (2015). A relax-and-repair heuristic for the swap-body vehicle routing problem. *Annals of Operations Research*, 1–22. doi:10.1007/s10479-015-2098-8.

Caramia, M., & Guerriero, F. (2010). A heuristic approach for the truck and trailer routing problem. *Journal of the Operational Research Society*, *61*(7), 1168–1180.

Chao, I.-M. (2002). A tabu search method for the truck and trailer routing problem. *Computers & Operations Research*, *29*, 33–51.

Derigs, U., Pullmann, M., & Vogel, U. (2013). Truck and trailer routing problems, heuristics and computational experience. *Computers & Operations Research*, *40*(2), 536–546.

Drexl, M. (2011). Branch-and-price and heuristic column generation for the generalized truck-and-trailer routing problem. *Journal of Quantitative Methods for Economics and Business Administration*, *12*(1), 5–38.

Drexl, M. (2013). Applications of the vehicle routing problem with trailers and transshipments. *European Journal of Operational Research*, *227*(2), 275–283.

Feo, T. A., & Resende, M. G. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, *6*(2), 109–133.

Gerdessen, J. C. (1996). Vehicle routing problem with trailers. *European Journal of Operational Research*, *93*(1), 135–147.

Gillett, B. E., & Miller, L. R. (1974). A heuristic algorithm for the vehicle-dispatch problem. *Operations Research*, *22*(2), 340–349.

Golden, B. L., Raghavan, S., & Wasil, E. A. (2008). *The vehicle routing problem: Latest advances and new challenges* (Vol. 43). New York: Springer.

Groer, C. (2012). VRPH. http://projects.coin-or.org/VRPH. Accessed May 2016.

Heid, W., Hasle, G., & Vigo, D. (2014). Verolog solver challenge 2014: Problem description. In *VeRoLog (EURO working group on vehicle routing and logistics optimization) and PTV group* (pp. 1–6).

Huber, S., & Geiger, M. J. (2014). Swap body vehicle routing problem: A heuristic solution approach. In *Computational Logistics* (pp. 16–30). Berlin: Springer.

Lin, S.-W., Yu, V. F., & Chou, S.-Y. (2009). Solving the truck and trailer routing problem based on a simulated annealing heuristic. *Computers & Operations Research*, *36*(5), 1683–1692.

Lin, S.-W., Yu, V. F., & Chou, S.-Y. (2010). A note on the truck and trailer routing problem. *Expert Systems with Applications*, *37*(1), 899–903.

Lum, O., Chen, P., Wang, X., Golden, B., & Wasil, E. (2015). A heuristic approach for the swap-body vehicle routing problem. In *14th INFORMS computing society conference* (pp. 172–187).

Open MP Architecture Review Board. (2011). *OpenMP application program interface version, version 3.1*.

Scheuerer, S. (2006). A tabu search heuristic for the truck and trailer routing problem. *Computers & Operations Research*, *33*(4), 894–909.

Semet, F. (1995). A two-phase algorithm for the partial accessibility constrained vehicle routing problem. *Annals of Operations Research*, *61*(1), 45–65.

Semet, F., & Taillard, E. (1993). Solving real-life vehicle routing problems efficiently using tabu search. *Annals of Operations Research*, *41*(4), 469–488.

Tan, K., Chew, Y., & Lee, L. (2006). A hybrid multi-objective evolutionary algorithm for solving truck and trailer vehicle routing problems. *European Journal of Operational Research*, *172*(3), 855–885.

Toth, P., Vigo, D., Toth, P., & Vigo, D. (2014). *Vehicle routing: Problems, methods, and applications* (2nd ed.). Philadelphia: Society for Industrial and Applied Mathematics.

Villegas, J. G., Prins, C., Prodhon, C., Medaglia, A. L., & Velasco, N. (2011). A GRASP with evolutionary path relinking for the truck and trailer routing problem. *Computers & Operations Research*, *38*(9), 1319–1334.