A relax-and-repair heuristic for the Swap-Body Vehicle Routing Problem

Nabil Absi, Diego Cattaruzza, Dominique Feillet, Sylvain Housseman

A relax-and-repair heuristic for the Swap-Body Vehicle Routing Problem

Nabil Absi[1], Diego Cattaruzza[2,1], Dominique Feillet[1], and Sylvain Housseman[1]

[1]Ecole des Mines de Saint-Etienne and LIMOS UMR CNRS 6158, CMP Georges Charpak, F-13541 Gardanne, France
[2]Univ. Lille, CNRS, Centrale Lille, UMR 9189 - CRIStAL - Centre de Recherche en Informatique Signal et Automatique de Lille, F-59000 Lille, France

Abstract

In this paper we address the Swap-Body Vehicle Routing Problem, a variant of the Truck and Trailer Routing Problem. It was introduced in the VeRoLog Challenge 2014. We develop a solution approach that we coin *Relax-and-Repair*. It consists in solving a relaxed version of the SB-VRP and deriving a feasible solution by repairing the relaxed one. We embed this approach within a population-based heuristic. During computation we store all feasible routes in order to derive better solutions by solving a set-partitioning problem. In order to take advantages of nowadays multi-core machines, our algorithm is designed as a collaborative parallel population-based heuristic. Experimental results show that our relax-and-repair algorithm is very competitive and point the impact of each phase on the quality of the obtained solutions. The advantage of our approach is that it can be adapted to solve complex industrial routing problems.

Keywords: vehicle routing; swap-body; genetic algorithm; relax-and-repair.

1 Introduction

In this paper we address the so called Swap-Body Vehicle Routing Problem (SB-VRP) introduced in the VeRoLog[1] Challenge 2014. This problem is a variant of the classic VRP. The goal of the challenge was to deal with realistic vehicle routing problems (VRP), and it was proposed in collaboration with the German company PTV[2].

The SB-VRP is a generalized version of the well known Truck and Trailer Routing Problem (TTRP). In the TTRP, the capacity of trucks can be increased by attaching a trailer. Using such configuration is however not allowed when visiting some customers. Furthermore, certain customer locations can be used to temporarily park a trailer in order to be able to visit customers with accessibility limitations.

In the SB-VRP the fleet is made by vehicles able to pull up to two *swap-bodies*. Comparing with the TTRP, the counterpart of a truck (resp. a trailer) is a vehicle pulling one swap-body (resp. two swap-bodies). The network includes special locations, called *swap-locations*, where

[1]EURO Working Group on Vehicle Routing and Logistics Optimization (`www.verolog.eu`)
[2]PTV is a German company operating worldwide that offers software, data, content, consulting and research (`www.ptvgroup.com`)

swap-bodies can be parked, picked-up or where their pulling order can be changed. This allows vehicles to leave the depot with two swap-bodies and be able to visit customers with restricted accessibility with each swap-body while the other is parked in a swap-location. This gives more flexibility during loading operations than in the TTRP.

We propose to tackle the SB-VRP with what we call a relax-and-repair procedure. The principle of this approach is: a. To identify a well known academic problem sharing strong similarities with the SB-VRP (the "relaxed problem"); b. To apply an existing solution approach for the solution of this problem; c. To transform the most promising solutions obtained into SB-VRP solutions by applying a repair procedure.

The motivation for this approach comes from the recent development of very efficient solution procedures for rich VRP. With a single black-box method, it is now possible to solve efficiently a large variety of vehicle routing problems (Cordeau *et al.* [3], Irnich [8], Vidal *et al.* [21]). However, these methods are still limited in their scope: while all combinations of classic components of routing problems can be considered, realistic problems like the SB-VRP are still out of reach (without entering into the core of the black-box). Relax-and-repair is a possible mechanism to simultaneously address difficult realistic problems and take advantage of powerful black-box methods.

In our implementation, the "relaxed problem" is the Heterogeneous-Fleet VRP. As recent methods able to solve efficiently this problem are not publicly available, we developed our own solution algorithm. Furthermore, because of the context of the challenge, the method presented subsequently is a little bit more complex than the pure relax-and-repair framework sketched above.

The paper is organized as follows. Section 2 formally defines the SB-VRP. Section 3 reviews the related literature. Sections 4 to 7 describe the algorithm. Section 8 presents the computational results and finally Section 9 concludes the paper.

2 Problem definition

The Swap-Body Vehicle Routing Problem (SB-VRP) can be defined on a complete directed graph $\mathcal{G} = (\mathcal{X}, \mathcal{A})$, where $\mathcal{X} = \{0, 1, \ldots, N, N+1, \ldots, N+S\}$ is the set of nodes. Node 0 represents the depot where the fleet is initially located. Nodes $1, \ldots, N$ represent the customers that require a service, and nodes $N+1, \ldots, N+S$ represent the S *swap-locations* available in the network. Swap-locations allow vehicles to perform particular operations described in the following. Let $\mathcal{X} = \mathcal{N} \cup \mathcal{S}$ where $\mathcal{N} = \{0, 1, \ldots, N\}$ and $\mathcal{S} = \{N+1, \ldots, N+S\}$. $\mathcal{A} = \{(i,j) | i, j \in \mathcal{X}\}$ is the set of arcs. It is possible to drive from one location $i \in \mathcal{X}$ to another location $j \in \mathcal{X}$, $i \neq j$ incurring in a driving time T_{ij} and covering a distance D_{ij}.

The fleet is initially based at the depot and it is composed by an infinite number of each of the following three elements: *truck*, *semi-trailer* and *swap-body*, the latter indicated by SB from now on. Only the SB has a strictly positive capacity Q. A truck can leave the depot carrying one SB, or can be attached to a semi-trailer and then carry two SB. In this case we will refer to it as a *train*. From now on, the world *truck* will refer to vehicles carrying only one SB. Figure 1 depicts these two vehicles.

Each customer i requires a quantity Q_i of products that has to be delivered. Service at its location takes ST_i units of time. The set of customers is partitioned in two subsets: $\mathcal{N}_t$ and $\mathcal{N}_T$. $\mathcal{N}_t$ contains all the customers that can be visited *only* by a truck, called *truck customers*. On the other side, $\mathcal{N}_T$ contains the customers that can be visited by either a train or a truck,

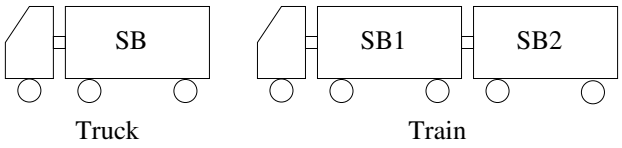


Figure 1: The SB-VRP vehicles

called *train customers*. Given a customer i, we define $TT(i) = 1$ if $i \in \mathcal{N}_T$, 0 otherwise.

We indicate the status of a vehicle, i.e., the SB it is carrying, with a pair $(\cdot, \cdot)$. This pair has to be red as if the vehicle was moving from right to left: the first value indicates the closest SB with respect to the driver, the second value the farthest. A truck is then represented by $(SB, \emptyset)$, where SB is the SB it is carrying.

Let now v be a train leaving the depot with configuration $(SB1, SB2)$, where $SB1$ and $SB2$ are two swap-bodies. v can perform several actions at each swap-location $s \in \mathcal{S}$, that are reported in Table 1 and Figure 2. In particular, v arrives at s in its *initial status* and leaves from s in its *final status*. Each action requires a manoeuvre time. As an example, the action *park* allows the train to let the second SB at a swap-location s. In particular, if it arrives at s in its status $(SB1, SB2)$ it will leave s as $(SB1, \emptyset)$. If the vehicle needs to continue its route with only the second SB (i.e., in the status $(SB2, \emptyset)$) it needs to perform an *exchange*. Note that different moves have different maneuver times which means that the sequence of the SB matters. Moreover, no switch of SB can be performed among different vehicles and each vehicle must carry back to the depot the SB he had when it left it.

action name	initial status	final status
park	$(SB1, SB2)$	$(SB1, \emptyset)$
pick-up	$(SB1, \emptyset)$	$(SB1, SB2)$
swap	$(SB1, \emptyset)$	$(SB2, \emptyset)$
exchange	$(SB1, SB2)$	$(SB2, \emptyset)$

Table 1: Actions at the swap-locations

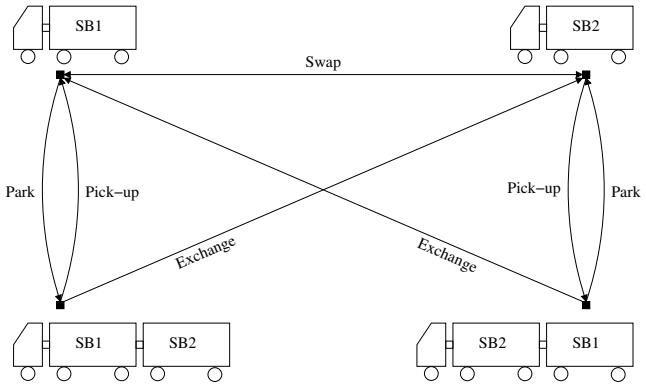


Figure 2: Actions at the swap-locations

Customers must be visited exactly once. However, the demand of a customer i can be

split between two different SB as long as these two SB are assigned to the same train and the train is complete (the two SB attached) when i is visited. A time horizon T_H is given and represents the maximum time a route can last. Due to the absence of time windows, the policy "start as soon as you can" is optimal and then T_H can be viewed as a limit on the duration of the working day.

Given a route r, let $T(r)$ be its total time (the sum of travel time, service time and maneuver time at swap-locations), $D(r)$ its traveled distance, $D^{train}(r)$ its traveled distance with the vehicle pulling two SB. When route r is performed by a truck $D^{train}(r) = 0$, while when a train does not visit swap-locations $D(r) = D^{train}(r)$. Let δ_r be an indicator equal to 1 when route r is performed by a train.

The objective is to find a solution ξ that minimizes the total operational cost $c(\xi)$:

$$c(\xi) = \sum_{r=1}^{R} \left(c_{fix}^{truck} + \delta_r c_{fix}^{train} + c_{hour}^{truck} T(r) + c_{km}^{truck} D(r) + \delta_r c_{km}^{train} D^{train}(r) \right). \quad (1)$$

where R is the number of routes of the solution.

In particular, the cost of each route $r \in R$ is the sum of three terms: a fixed cost $c_{fix}^{truck} + \delta_r c_{fix}^{train}$ for using the truck and possibly the trailer with the second SB, a cost $c_{hour}^{truck} T(r)$ depending on the time needed to perform r, plus a cost $c_{km}^{truck} D(r) + \delta_r c_{km}^{train} D^{train}(r)$ depending on the traveled distance $D(r)$ and the possible distance $D^{train}(r)$ where the second SB is pulled.

3 Literature review

The SB-VRP can be seen as a variant of the Truck and Trailer Routing Problem (TTRP) in which the set of customers is partitioned into two subsets. A first set of customers that can only be visited by a truck (truck customers), and a second set of customers that can be visited by either a truck or a truck pulling a trailer (trailer customers). The capacity of each truck and each trailer is given and the number of trucks and trailers is limited. A trailer can be dropped-off at the depot or at a trailer customer location but should be picked-up and returned to the depot. Maneuvering time is needed to access customers to drop-off a trailer. The main objective of the TTRP is to find a set of routes that visits all clients, satisfies the customers accessibility constraints and minimizes the total cost associated with total traveled distance of trucks and trailers. The main difference between the SB-VRP and TTRP is that the SB-VRP considers the presence of swap-locations where more operations than dropping-off and picking up the trailer are possible (see Table 1). This implies much more complicated routes. In addition, the objective function of the SB-VRP is richer than the TTRP one (see Equation 1).

To the best of our knowledge, Semet and Taillard [17] is the first paper that deals with the TTRP. The authors address an industrial problem that deals with the routing of trucks and trailers with customer time windows. They consider a heterogeneous fleet of vehicles, their related customer accessibility constraints and access times. The authors develop a tabu search algorithm to tackle this problem. Gerdessen [6] addresses a special version of the TTRP: each trailer is parked exactly once and all customers have unit demand. These two assumptions considerably simplify the problem. In fact, the first assumption makes the feasibility problem easy to check and the second limits the number of possible routes. The objective function is

the sum of the time that vehicles need to cover the routes and the total access time. They use a two-phase heuristic based on a construction phase and an improvement phase.

In Chao [2], Scheuerer [15] and Lin *et al.* [9] the routing costs do not differentiate between tours with and without trailers. The objective function is based on the total traveled distance. It does not consider any other cost components like fixed costs, costs for shifting demand, or costs for coupling or uncoupling of trailers. Chao [2] and Scheuerer [15] propose tabu search algorithms, while Lin *et al.* [9] develops a very effective simulating annealing algorithm that performs better than previous studies. In Lin *et al.* [10], the same authors use the same simulating annealing algorithm to address a version of the TTRP in which the truck and trailer availability constraints are relaxed. Villegas *et al.* [23] proposes a hybrid metaheuristics based on a greedy randomized adaptive search procedure (GRASP), a variable neighborhood search and a path relinking. Numerical results show that their methods outperform all previously published methods. Other variants of the TTRP were addressed in the literature. Semet [16] addresses a similar problem called the partial accessibility constrained vehicle routing problem (PACVRP). In this problem, each parking place for the trailer is restricted to have only one subtour. To solve the problem, the author develops a cluster-first route-second procedure. Villegas *et al.* [22] studies the single truck and trailer routing problem with satellite depots where goods can be transferred between the trucks and the trailers. To solve the problem, the authors propose a multi-start evolutionary local search and a hybrid metaheuristic based on GRASP and variable neighborhood descent paradigms. Drexl [4] describes how several important types of vehicle routing problems with multiple synchronization constraints such as multi-echelon location-routing problems and simultaneous vehicle and crew routing problems, can be modeled as VRPTT.

The literature on the SB-VRP is limited to a few recent papers from participants to the challenge. In Huber and Geiger [7], the authors propose an iterated variable neighborhood search to solve the problem. They represent a vehicle route as a set of successive segments, indicating the changes of status for the vehicle (i.e., visits to a swap location or the depot). In addition to classical local search operators, they introduce an intra-route operator moving customers between different segments of a route. Also, they introduce an operator modifying a swap location. When a local optimum is obtained, they perturb the solution by destroying some routes and reinserting the customers.

In Lum *et al.* [11], the authors start by considering only trucks: vehicles always travel with a single swap-body and swap locations are ignored. They build a solution with an off-the-shelf VRP library, slightly modified to take care of the complex cost function. Customers with a demand greater than the swap-body capacity are not included. They are added in post-processing with single-customer train routes. Note that the special case where all customers can be visited by trains (see Section 8) is addressed with the same off-the-shelf library but only considering trains; a quick post-processing is then applied to transform train routes into truck routes when the capacity of a single swap-body is exploited. The solution is improved with a Variable Neighborhood Descent procedure. Classical local search operators are applied to truck routes or to the main tour of train routes (the one traveled with the two swap-bodies). A so-called migration operator is added to move truck customers from truck routes into subtours on train routes. When a local optimum is obtained, the solution is perturbed with a swap-string operation.

In Miranda-Bront *et al.* [13], the authors propose a GRASP procedure. In the construction phase, the type of route - truck or train - is randomly chosen when new routes are opened. Customer insertion admits complex insertions such as the insertion of the customer surrounded

by visits to a swap-location. An Iterated Local Search approach is used for improvement. Local search is performed with a set of classical and specialized operators. The perturbation operator used when a local optimum is obtained, randomly selects a subset of customers, that are removed and reintroduced. In order to improve the efficiency of the whole scheme, the instance is first decomposed with a cluster-first mechanism and the GRASP method is applied to each cluster independently. After a given amount of time, the best solutions found in the different clusters are merged and the Iterated Local Search rerun until completion.

Computational results obtained in these three papers are discussed in Section 8.

4 Algorithm outline

The algorithm is based on a two-phase paradigm. A Heterogeneous-Fleet VRP (H-VRP) is first solved, where two types of vehicles are available: vehicles of type v_1 with a capacity equal to one SB, that is Q, and vehicles of type v_2 with a capacity that is twice the capacity of a SB, that is $2Q$. Routing a vehicle of type v_1 costs exactly as routing a truck, while routing a vehicle of type v_2 costs as much as routing a train. Precisely, if route r is performed by a vehicle of type v_1, the cost is

$$C(r)^{v_1} = C(r)^{truck} = c_{fix}^{truck} + c_{hour}^{truck} T(r) + c_{km}^{truck} D(r) \tag{2}$$

while it costs

$$C(r)^{v_2} = C(r)^{train} = c_{fix}^{truck} + c_{fix}^{train} + c_{hour}^{truck} T(r) + (c_{km}^{truck} + c_{km}^{train}) D(r) \tag{3}$$

if r is performed by a vehicle of type v_2.

In this phase, accessibility constraints are not considered: each customer can be visited by either a truck or a train. As a consequence, the solution obtained when solving the H-VRP, can be infeasible for the SB-VRP. In this case, the second phase of the algorithm is launched in order to *repair* the solution (Section 6). The *repairing procedure* considers all the routes in the H-VRP solution that are infeasible for the SB-VRP. These routes are performed by vehicles of type v_2. Visits to swap-locations are added and the customer demand assigned to the two SB, in order to make the solution feasible with a minimal cost increase. It may happen that the reparation is impossible. Delays caused by the insertion of swap-locations and by the associated operations can be incompatible with the time horizon. In addition, the assignment of customers to a specific SB can be impossible. In this case the solution is penalized (i.e., its cost is increased).

We decided to tackle the solution of the H-VRP with a memetic algorithm following the concepts introduced in Vidal *et al.* [20], which demonstrated their efficiency for many classes of vehicle routing problems. Additionally, to take advantages from multi-core architecture in nowadays computer processors, we propose a collaborative parallel population-based procedure to solve the SB-VRP (Section 5.2). On each thread a population-based algorithm is run. Populations are formed by chromosomes (Section 5.1 explains the chromosome encoding). Each chromosome is a representation of a H-VRP solution. Children chromosomes are created from parents chromosomes already present in the population in the hope that children will be built up on the *good* bricks of the parents (details are given in Section 5.4).

Finally, the algorithm includes a post-optimization phase that aims at improving the best obtained solution. To this purpose, all the feasible routes created during the search are stored

into a pool $\mathcal{R}$. At the end of the collaborative procedure, a set-partitioning problem on the set of customers $\mathcal{N}$ is solved in order to find a new best solution (Section 7). The set partitioning procedure can also be solved during the collaborative procedure in order to enrich the current population. The concept of the method is summarized in Figure 3.

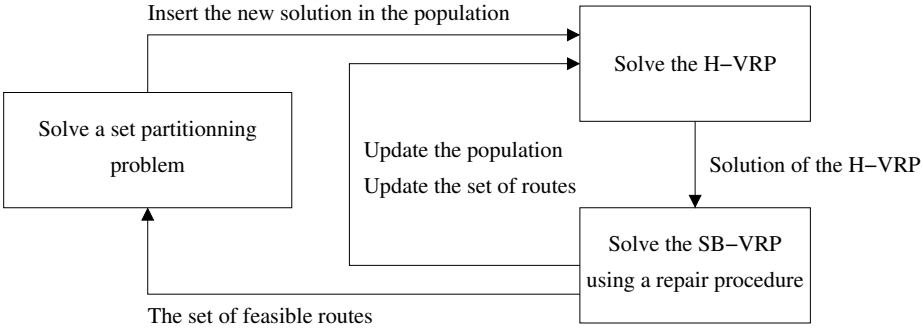


Figure 3: The solution algorithm

5 Population algorithm for the H-VRP

5.1 Chromosome encoding

A chromosome $\Psi = (\Psi_1, \ldots, \Psi_N)$ is simply a permutation of the N customers to serve. In the literature it is usually called *giant-tour*. An initial H-VRP solution is obtained from each chromosome by means of a *split* procedure (see Section 5.5), inspired by the procedure used in Prins [14] to turn a chromosome into a VRP solution. Note that swap-locations are not represented in the chromosome.

5.2 Collaborative population based algorithm

In order to take the advantages of parallel computing available in nowadays computers, a collaborative procedure was implemented. In particular, on each thread, the same population-based procedure is run (see Section 5.4). In this section we explain the general structure of the collaborative procedure.

On each thread h of the n_{thread} available threads, a population $\mathcal{P}_h$ of chromosomes is kept. Each $\mathcal{P}_h$ is first initialized (see Section 5.3) with n_{init} chromosomes. Each population $\mathcal{P}_h$ is then independently managed as explained in Section 5.4. The populations do not directly communicate with each other, but rather independently evolve until each population contains n_{final} chromosomes.

The chromosomes of all populations $\mathcal{P}_h$ are then inserted into a common empty population $\mathcal{P}$. On $\mathcal{P}$, a *survivor* procedure is launched. In particular, the survivor procedure consists in evaluating the chromosomes according to a *biased fitness* BF, as proposed in Vidal *et al.* [20], that takes into consideration the chromosome quality (proportional to the cost of its associated SB-VRP solution) and their contribution to the diversification of the population (to avoid premature convergence). After the survivor procedure is performed, chromosomes in $\mathcal{P}$ are ordered with respect to the BF.

All the populations $\mathcal{P}_h$ are re-initialized with the best $\frac{2}{3}n_{init}$ chromosomes with respect to the BF. Then, each of the following $\frac{n_{thread}n_{init}}{3}$ chromosomes is randomly inserted in one of

the populations. This prevents the populations from being identical. With a probability p_{mut} a new chromosome is generated and inserted into all populations. The procedure is repeated until the time limit $t_{collaborative}$ is reached.

Each time a new best solution for the SB-VRP is found, it is memorized into a global variable (shared by all the parallel threads).

A pool $\mathcal{R}_h$ of routes is associated with each thread. It stores all feasible routes (for the SB-VRP) that are obtained during the search.

A similar procedure can be found in Duhamel *et al.* [5] in the context of the H-VRP.

5.3 Initial population

On each of the n_{thread} threads, the population $\mathcal{P}_h$ is initialized with n_{init} chromosomes. The initial permutations are constructed as follows. A customer is randomly selected and a route that uses a vehicle of type v_1 is initialized with it. Next customer i is randomly selected and inserted between customers j and k that minimize the following saving cost (Vaz Penna *et al.* [19]):

$$c_{ji} + c_{ik} - c_{jk} - c_{0i} - c_{i0}. \tag{4}$$

If the insertion of customer i in a route performed by a vehicle of type v_1 violates the capacity, the route is considered to be performed by a vehicle of type v_2. Moreover, the possibility of initializing a new route with only customer i is considered. In the latter case, fixed costs of using a vehicle are counted. Costs involved in Equation (4) take into account both variable costs related to traveled time and distance arising from routing a vehicle of type v_1 or v_2, depending on the current route. The terms in Equation (4) involving the depot allow to avoid late insertions of customers located far away from the depot.

Each permutation is transformed into a H-VRP solution using a *split* procedure described in Section 5.5. The obtained solutions are improved applying a local search procedure described in Section 5.6. A SB-VRP solution is obtained by repairing a H-VRP solution (Section 6). This part correspond to lines 6–15 of Algorithm 1 and are explained in details in Section 5.4.

5.4 Population-based procedure

This section explains the management of each population $\mathcal{P}_h$ assigned to a particular thread. A sketch of the algorithm is given in Algorithm 1.

The algorithm starts by initializing an empty pool of feasible routes $\mathcal{R}$. At each iteration, two chromosomes Ψ_1 and Ψ_2 are selected from the population. Best chromosomes have a higher probability to be selected. The second chromosome Ψ_2 is selected in the population among those with a different cost than Ψ_1 (lines 3–4). This is done in order to avoid to select twice the same chromosome. The classic order crossover operator OX is used to obtain a child Ψ from the parents Ψ_1 and Ψ_2 (line 5). The new chromosome undergoes the *split* procedure (line 6) to obtain a feasible solution ξ^{H-VRP} for the H-VRP. ξ^{H-VRP} is improved by means of a local search procedure (line 7) and possibly repaired (line 8) in order to obtain a feasible solution ξ^{SB-VRP} for the SB-VRP. If the resulting solution ξ^{SB-VRP} is infeasible, its cost is multiplied by a coefficient θ (lines 9–13), otherwise all feasible routes are inserted into the pool $\mathcal{R}$. The routes are then concatenated and the permutation of customers updated accordingly (line 14). This allows the chromosome to well represent the structure of the solution. The chromosome is then inserted into the population (line 15). The procedure is repeated until

Algorithm 1 Population-based algorithm outline

$\mathcal{R} = \emptyset$
while $|\mathcal{P}_h| < n_{final}$ **do**
  $\Psi_1 = Select(\mathcal{P}_h)$
  $\Psi_2 = Select(\mathcal{P}_h, \Psi_1)$
  $\Psi = OX(\Psi_1, \Psi_2)$
  $\xi^{H-VRP} = Split(\Psi)$
  $\xi^{H-VRP} = LS(\xi^{H-VRP})$
  $\xi^{SB-VRP} = Repair(\xi^{H-VRP})$
  if ξ^{SB-VRP} is infeasible **then**
    $c(\xi^{SB-VRP}) = \theta c(\xi^{SB-VRP})$
  else
    $\mathcal{R} = \mathcal{R} \cup \{\text{all routes of } \xi^{SB-VRP}\}$
  end if
  $\Psi = Concatenate(\xi^{SB-VRP})$
  $Insert(\mathcal{P}_h, \Psi)$
end while

the population reaches a size equal to n_{final}. This same procedure is run on each of the available threads.

5.5 Split procedure

The *split* procedure constructs a H-VRP solution ξ from a chromosome Ψ. It does not consider the swap-locations.

Split works on an auxiliary graph $\mathcal{G}' = (\mathcal{N}', \mathcal{A}')$. $\mathcal{N}'$ contains $N+1$ nodes, indexed from 0 to N. $\mathcal{A}'$ contains arc (i, j) if it is feasible to perform route $r = (0, \Psi_{i+1}, \ldots, \Psi_j, 0)$ either with a vehicle of type v_1 or with a vehicle of type v_2. The costs on the arcs are fixed according to the following proposition.

Proposition 1. *Given a route r such that $Q(r) \leq Q$, it is never beneficial to perform it with a vehicle of type v_2.*

Proof. Let us recall that a vehicle of type v_1 in the H-VRP context is equivalent to a truck in the SB-VRP context, while routing a vehicle of type v_2 costs as much as routing a train along the whole route (the two SB are always pulled and swap-locations are never visited). Given a route r let $C(r)^{v_1} = C(r)^{truck}$ and $C(r)^{v_2} = C(r)^{train}$ be respectively the cost of performing the route r with a vehicle of type v_1 and with a vehicle of type v_2. Let suppose that $Q(r) \leq Q$. We know from Equations (2) and (3) that:

$$C(r)^{v_1} = C(r)^{truck} = c_{fix}^{truck} + c_{hour}^{truck} T(r) + c_{km}^{truck} D(r),$$

and

$$C(r)^{v_2} = C(r)^{train} = c_{fix}^{truck} + c_{fix}^{trailer} + c_{hour}^{truck} T(r) + (c_{km}^{truck} + c_{km}^{trailer}) D(r).$$

Since the coefficients are non negative, $C(r)^{truck} \leq C(r)^{trailer}$. This concludes the proof. □

This proposition implies that it is never beneficial to use a vehicle of type v_2 to perform a route $r = (\Psi_{i+1}, \ldots, \Psi_j)$, when $Q(r) \leq Q$. The costs on the arcs (i, j) are then fixed as follows: $c_{ij} = C(r)^{v_1}$, if $Q(r) \leq Q$, and $c_{ij} = C(r)^{v_2}$ otherwise.

Due to the infinite availability of each fleet component, it is only needed to compute the shortest path on the auxiliary graph $\mathcal{G}'$ (otherwise computing a shortest path with resource constraints would be needed). Then, to each arc (that represents a route) that belongs to the shortest path is assigned a vehicle of type v_1 or of type v_2 according to the transported quantity of merchandise.

5.6 Local search

The local search procedure is applied to the solution that is obtained by the *split* procedure. It is thus applied to a H-VRP solution and does not consider the swap-locations. Classical moves used in routing context can be used.

Let $i, j \in \mathcal{N}$ be two customer nodes. Let $\sigma(i)$ and $\sigma(j)$ be their successors. Let $r(i)$ and $r(j)$ indicate their respective routes. The local search procedure is based on the following moves:

- relocate i after j;
- if $\sigma(i)$ is not the depot, relocate $(i, \sigma(i))$ after j;
- if $\sigma(i)$ is not the depot, remove $(i, \sigma(i))$ from $r(i)$ and insert $(\sigma(i), i)$ after j;
- exchange i with j;
- if $\sigma(i)$ is not the depot exchange $(i, \sigma(i))$ with j;
- if $\sigma(i)$ is not the depot replace $(i, \sigma(i))$ with j, and j with $(\sigma(i), i)$;
- if $\sigma(i)$ and $\sigma(j)$ are not the depot exchange $(i, \sigma(i))$ with $(j, \sigma(j))$;
- if $\sigma(i)$ and $\sigma(j)$ are not the depot replace $(i, \sigma(i))$ with $(j, \sigma(j))$ and $(j, \sigma(j))$ with $(\sigma(i), i)$;
- if $r(i) = r(j)$, replace $(i, \sigma(i))$ and $(j, \sigma(j))$ with (i, j) and $(\sigma(i), \sigma(j))$;
- if $r(i) \neq r(j)$, replace $(i, \sigma(i))$ and $(j, \sigma(j))$ with $(i, \sigma(j))$ and $(j, \sigma(i))$;

Note that the last two moves are the classical 2-opt and 2-opt*. The neighborhoods defined by each one of the previous moves are progressively explored one after the other. Before exploring a new neighborhood, the best improving move, if any, is performed. An improving move is a move that decreases the cost of the current solution. The local search stops when all the neighborhoods are explored in turn, without finding an improving move.

To reduce the size of the neighborhoods, a granular local search is implemented (Toth and Vigo [18]): a move is considered only if j is among the $n_{closest}$ closest customers of i. The $n_{closest}$ closest customers of each customer can be calculated in the preprocessing phase.

6 Repair procedure

If the SB-VRP instance contains at least one customer that cannot be visited by a train, the solution ξ obtained by solving the H-VRP can be infeasible. In fact, the route r serving this customer can be assigned to a vehicle of type v_2. The implementation of this solution in the SB-VRP context would require a train to perform r, violating the accessibility constraints.

The *repair* procedure aims at repairing these violations, and then considers all routes made by vehicles of type v_2 in which at least one of the visited customer is a truck customer. The routes made by truck are by definition feasible. The labeling procedure explained in Section 6.2 repairs the routes individually by

- inserting swap-locations;
- assigning demands to swap-bodies.

For each route r of the solution ξ of the H-VRP that would be infeasible in the SB-VRP context, the procedure constructs an acyclic graph whose nodes are the customers visited in r. Details on the graph construction are given in Section 6.1. The labeling procedure (Section 6.2) determines a path on the graph which indicates how the swap-locations can be included in the route to make it feasible with a minimum additional cost. Because of some approximations in this procedure, a correction procedure (Section 6.3) finally determines the exact itinerary and cost of the route.

6.1 Graph construction

Given a route $r = (0, r_1, \ldots, r_R, 0)$ a graph $\mathcal{G}'' = (\mathcal{N}'', \mathcal{A}'')$ is constructed as follows. $\mathcal{N}'' = \{0, 1, \ldots, R\}$ and $\mathcal{A}'' = \mathcal{A}''_1 \cup \mathcal{A}''_2$ where:

$$\mathcal{A}''_1 = \left\{ (i,j) \;\middle|\; \begin{array}{l} i,j \in \mathcal{N}'', i < j \\ \sum_{k=i+1}^{j} Q_{r_k} \leq Q \\ \exists k = i+1, \ldots, j \text{ s.t. } TT(r_k) = 0 \end{array} \right\}$$

and

$$\mathcal{A}''_2 = \left\{ (i,j) \;\middle|\; \begin{array}{l} i,j \in \mathcal{N}'', i < j \\ \forall k = i+1, \ldots, j \text{ s.t. } TT(r_k) = 1 \end{array} \right\}$$

An arc (i,j) represents the fragment $(r_{i+1}, \ldots, r_j)$ of route r. Set $\mathcal{A}''_1$ contains arcs representing fragments of routes that visit at least one truck customer. These fragments cannot be performed in the SB-VRP context without including a visit to a swap-location. In the subsequent labeling procedure, if a path traverses an arc $(i,j) \in \mathcal{A}''_1$, it means that the train goes to a certain swap-location s before visiting r_{i+1}, where it leaves one of its two SB, and that it goes back to s after having visited r_j, to pick up the SB. We denote the corresponding extra routing cost δ_{ij} and associate it with the arc (i,j). Is is evaluated as follows,

$$\delta_{ij} = \min_{s \in \mathcal{S}} \{ c^{train}_{r_i s} + c^{truck}_{s, r_{i+1}} + c^{truck}_{r_j s} + c^{train}_{s, r_{j+1}} \} - c^{train}_{r_i, r_{i+1}} - c^{train}_{r_j, r_{j+1}} + C^{truck}_{r_{i+1}, r_j}, \; \forall (i,j) \in \mathcal{A}''_1, \quad (5)$$

where:

$$\begin{aligned} C^{truck}_{r_{i+1}, r_j} &= \sum_{k=i+1}^{j-1} (c^{truck}_{km} - c^{train}_{km}) D_{r_k r_{k+1}}; \\ c^{train}_{r_i s} &= (c^{truck}_{km} + c^{train}_{km}) D_{r_i s} + c^{truck}_{hour} T_{r_i s}; \\ c^{truck}_{s r_i} &= c^{truck}_{km} D_{s r_i} + c^{truck}_{hour} T_{r_i s}; \\ c^{train}_{r_i, r_{i+1}} &= (c^{truck}_{km} + c^{train}_{km}) D_{r_i, r_{i+1}} + c^{truck}_{hour} T_{r_i, r_{i+1}}. \end{aligned}$$

Other terms are calculated equivalently.

Note that Equation (5) implicitly associates a swap-location $s^*_{i,j}$ with each arc (i,j) in $\mathcal{A}_1^{''}$.

Most importantly, note also that the evaluation of the extra routing cost δ_{ij} implicitly assumes that customers r_i and r_{j+1} are visited with trains. If r_i or r_{j+1} is visited with a truck, it only gives an approximation.

Set $\mathcal{A}_2^{''}$ contains arcs that represent fragments visiting only trailer customers. These fragments can be performed by the train following the original route calculated for the vehicle of type v_2. No additional cost is associated with these arcs, i.e., $\delta_{ij} = 0$ for all $(i,j) \in \mathcal{A}_2^{''}$.

6.2 Labeling procedure

Given the route $r = (0, r_1, \ldots, r_R, 0)$ and the graph $\mathcal{G}^{''} = (\mathcal{N}^{''}, \mathcal{A}^{''})$ computed as explained in Section 6.1, a labeling procedure is run in order to determine which and where swap-locations need to be inserted in r to make it feasible. Moreover, the extra time needed at a swap-location to perform actions is calculated, depending on the current status of the vehicle. In the meantime, this procedure assigns the demands to swap-bodies. More precisely, a label $\mathcal{L}$ is a vector with six components, $\mathcal{L} = (q_{SB1},\ q_{SB2},\ t,\ status,\ \Delta c,\ pred)$, where

- $\mathcal{L}^1 = q_{SB1}$ is the quantity assigned to SB1;
- $\mathcal{L}^2 = q_{SB2}$ is the quantity assigned to SB2;
- $\mathcal{L}^3 = t$ is the additional time needed for the digressions (equivalent to the additional times evaluated in the computation of Equation (5)) plus the time at the swap-locations to perform the correct actions (that depends on the status of the vehicle);
- $\mathcal{L}^4 = status$ records the current status of the vehicle. It is needed to select the proper action to perform at the swap-location and the time it takes;
- $\mathcal{L}^5 = \Delta c$ records incurred additional cost, given as the sum of costs calculated by Equation (5) and the costs due to the actions performed at the swap location;
- $\mathcal{L}^6 = pred$ records the predecessor node in $\mathcal{G}^{''}$.

The procedure starts with the null label associated with node 0 in $\mathcal{G}^{''}$, and determines a path to reach node R by extending $\mathcal{L}$ to subsequent nodes across arc selection. A label $\mathcal{L}_i$ is extended to a label $\mathcal{L}_j$ considering an arc $(i,j) \in \mathcal{A}^{''}$. Two cases have to be considered:

- **1st case:** $(i,j) \in \mathcal{A}_1^{''}$. At least one of the customers $r_{i+1}, \ldots, r_j$ cannot be visited by a train. A swap-location s is visited before serving r_{i+1} and after r_j. The demand required by these customers need to be assigned to one SB. Let suppose it is the first. The label $\mathcal{L}_j$ is obtained as follows:

$$\mathcal{L}_j^1 = \mathcal{L}_i^1 + \sum_{k \in \{i+1,\ldots,j\}} Q_{r_k}$$
$$\mathcal{L}_j^2 = \mathcal{L}_i^2$$
$$\mathcal{L}_j^3 = \mathcal{L}_i^3 + t_{ij} + t_s$$
$$\mathcal{L}_j^4 = new\_status$$
$$\mathcal{L}_j^5 = \mathcal{L}_i^5 + \delta_{ij}$$
$$\mathcal{L}_j^6 = i$$

Value t_s represents the additional time needed to perform the actions at the swap-location s (before going to r_{i+1} and after the service at r_j). This value cannot be considered a priori when calculating the arc costs since it depends on the status of the vehicle, that in turn depends on the selected arcs in $\mathcal{G}''$. Value t_{ij} represents the additional traveling time due to the selection of arc (i, j). When the selected SB is the second, a similar case occurs. The two possibilities are considered and two labels are then created. When the capacity of a SB is exceeded ($q_{SB_1} > Q$ or $q_{SB_2} > Q$) or the time horizon length violated ($\mathcal{L}_j^3 > T_H - T_r^{H-VRP}$), the (partial) solution represented by the corresponding label is infeasible and the label is dropped.

- **2nd case:** $(i, j) \in \mathcal{A}_2''$. Customers $r_{i+1}, \ldots, r_j$ are visited by the train. Since the demands of the visited customers can be split between the two SB, there is no need to assign quantities to a particular SB at this moment. Moreover, the train follows the sequence given by the H-VRP solution. The status of the train does not change. The labels are updated as follows:

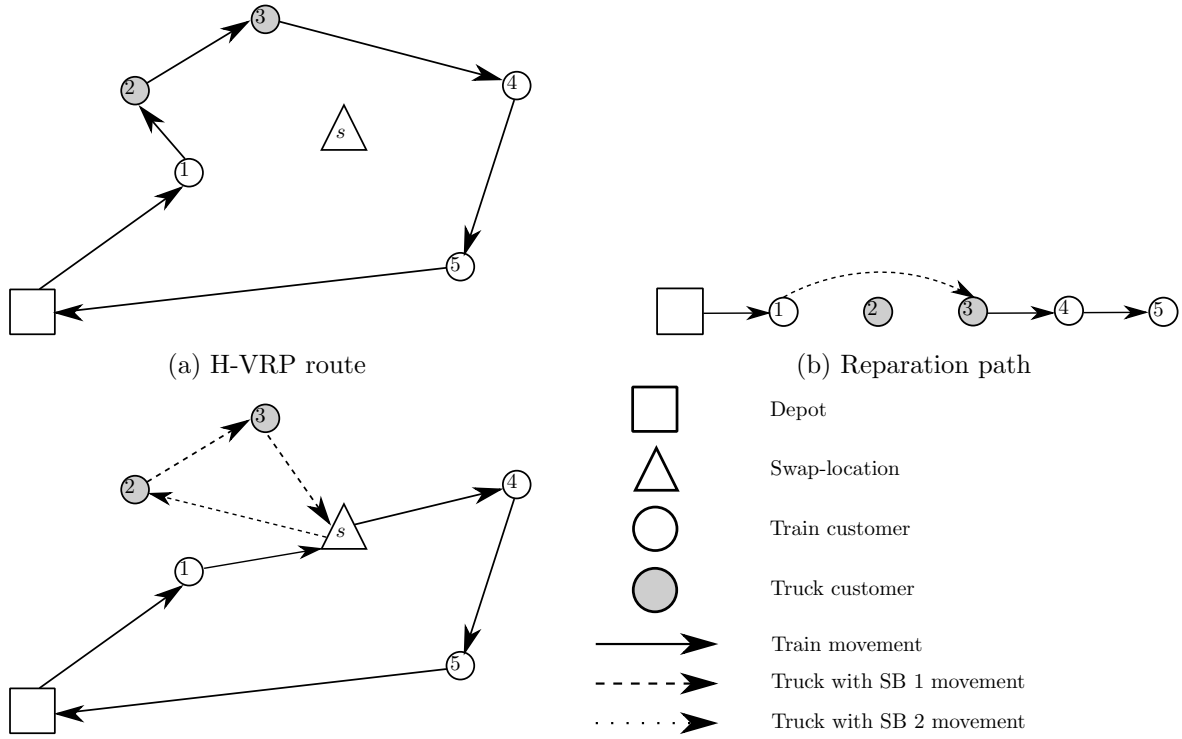
$$\mathcal{L}_j^h = \mathcal{L}_i^h \quad h = 1, \ldots, 5$$
$$\mathcal{L}_j^6 = i$$

Labels are extended until node R is reached and a path, called *reparation path*, from node 0 is determined. The minimum-cost label associated with node R, estimates the cost of the reparation. The reparation is determined sweeping backward the path.

6.3 Correction procedure

The labeling procedure determines a solution for the SB-VRP and an estimation of its cost. This cost only gives an exact evaluation when arcs in $(i, j) \in \mathcal{A}_1''$ are followed by arcs in $(j, k) \in \mathcal{A}_2''$. In this particular case, the vehicle goes to the swap-location s determined by arc (i, j), where it leaves one SB, visits customers $r_{i+i}, \ldots, r_j$, goes back to s to retrieve the SB, then goes to r_{j+1} before restarting the original planned route. The case is depicted in Figures 4a–4c for a simple route with five customers. Let us suppose that the total quantity required by the five customers is strictly bigger than Q and then the route is performed by a vehicle of type v_2 (this assumption will hold in successive examples). Figure 4a represents the base H-VRP route that is infeasible and needs to be repaired: customers 2 and 3 are truck customers (grey circles). Figure 4b represents the reparation path. Arc (i, j) is the arc

$(1,3)$, while $(j,k) = (3,4)$. Figure 4c represents the final route, that is feasible in the SB-VRP context. Figure 4d is the key for these figures (and for Figures 5 and 6).



(a) H-VRP route

(b) Reparation path

(c) SB-VRP route

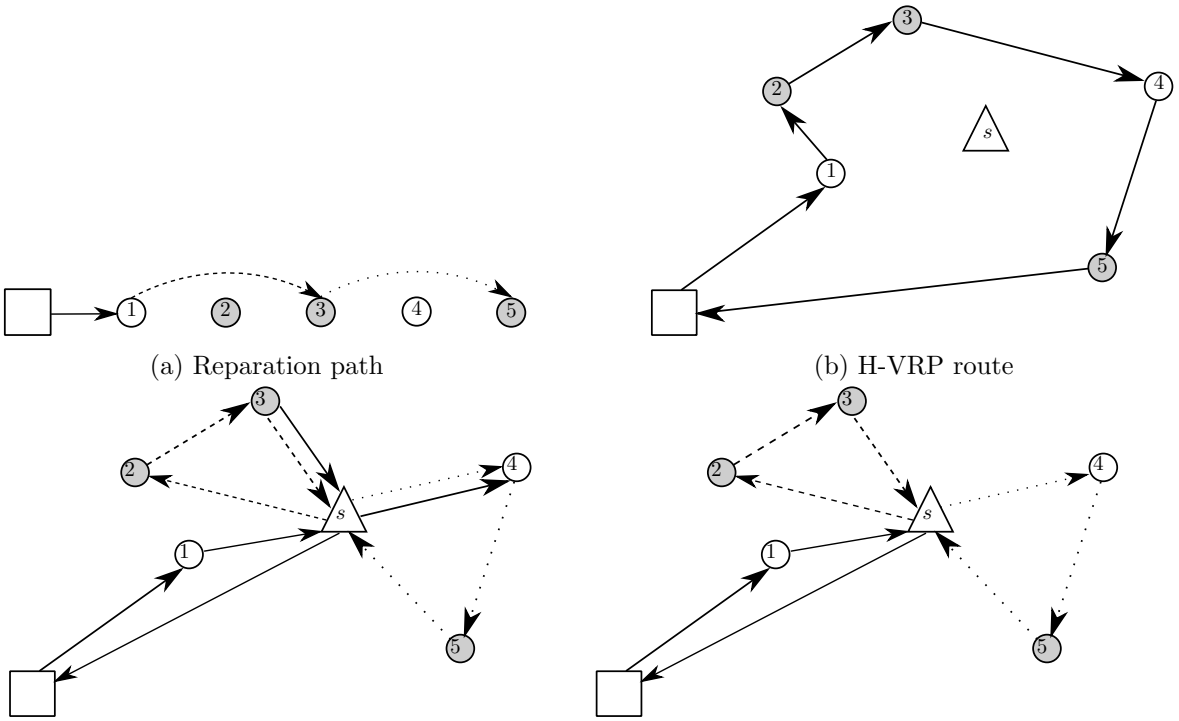
(d) Key

Figure 4: Arc $(i,j) \in \mathcal{A}_1''$ is followed by an arc $(j,k) \in \mathcal{A}_2''$

In the case that arc $(i,j) \in \mathcal{A}_1''$ is followed by another arc $(j,k) \in \mathcal{A}_1''$, a final correction procedure is needed to obtain the exact cost of the solution and to determine its feasibility in the SB-VRP context.

In particular, cost δ_{ij} takes into account the detour from r_i to a given swap-location s_1 before visiting customer r_{i+1}. The vehicle then serves customers $r_{i+1}, \ldots, r_j$ and goes back to s_1 to get back the SB before heading to r_{j+1}. Analogously, δ_{jk} takes into account the cost of the detour of visiting a swap-location s_2 between the visits of r_j and r_{j+1} and between r_k and r_{k+1}. Two cases appear.

1st case: $s_1 = s_2 = s$. The detours needed to go from r_j to s and from s to r_{j+1} are counted twice: the fist due to the arc (i,j) entering in j, the second due to the arc (j,k) leaving j. The cost of the route then needs to be updated to take this into account. Figures 5a–5d represent the case. Arc (i,j) is the arc $(1,3)$, while arc (j,k) is the arc $(3,5)$ (Figure 5a).



(a) Reparation path

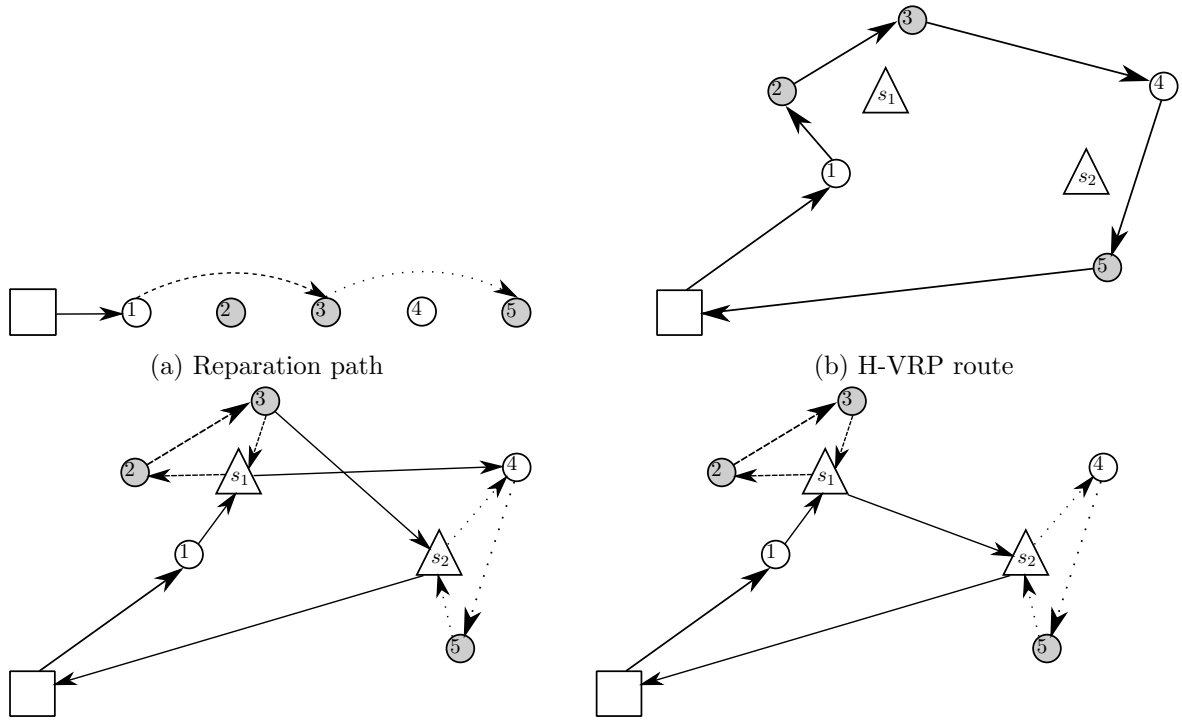
(b) H-VRP route

(c) SB-VRP intermediate route

(d) SB-VRP route

Figure 5: Arc $(i, j) \in \mathcal{A}_1''$ is followed by an arc $(j, k) \in \mathcal{A}_1''$ and $s_1 = s_2 = s$

2nd case: $s_1 \neq s_2$. In this case, arc (i, j) selects s_1, while arc (j, k) selects s_2. The selection of arc (i, j) imposes the visit of s_1 after r_j and before r_{j+1}. Arc (j, k) imposes the vehicle to go from r_j to s_2 and then to r_{j+1}. Finally, arc (i, j) imposes the vehicle to go across (r_j, s_1) and (s_1, r_{j+1}), while arc (j, k) to go across (r_j, s_2) and (s_2, r_{j+1}). To solve this problem we force the vehicle to travel directly from s_1 to s_2 and we update the cost of the route accordingly. Figures 6a–6d represent this case: arc $(i, j) = (1, 3)$ and arc $(j, k) = (3, 5)$.



(a) Reparation path

(b) H-VRP route

(c) SB-VRP intermediate route

(d) SB-VRP route

Figure 6: Arc $(i, j) \in \mathcal{A}_1^{''}$ is followed by an arc $(j, k) \in \mathcal{A}_1^{''}$ and $s_1 \neq s_2$

7 Post-optimization phase

As mentioned in Section 4, we implemented a post-optimization procedure to improve the best solution obtained by the collaborative procedure. This is done by keeping all feasible routes created during the population based procedure in a pool $\mathcal{R}$. We then try to find the best packing of a subset of these routes in order to improve the current best solution. Practically, this is done by solving a set partitioning problem (as done for example by Mendoza and Villegas [12]). After $t_{collaborative}$ units of time, the collaborative population-based procedure is stopped. All the routes stored in the pools $\mathcal{R}_h$ associated with each thread h are inserted into a common pool $\mathcal{R}$.

The set partitioning problem is defined by a set of routes $\mathcal{R}$ and the set of customers $\mathcal{N}$. For each route $r \in \mathcal{R}$, c_r is its cost and a_{ir} is a coefficient equal to 1 if customer i is visited by route r, 0 otherwise. Variable ζ_r is a binary variable that equals 1 if route r is selected, 0 otherwise. In the case that $\mathcal{R}$ enumerates all the feasible routes, the optimal solution for the SB-VRP could be obtained by solving the following set partitioning problem:

$$\min_{r \in \mathcal{R}} c_r \zeta_r \tag{6}$$

$$\sum_{r \in \mathcal{R}} a_{ir} \zeta_r = 1, \quad \forall i = 1, \ldots, N \tag{7}$$

$$\zeta_r \in \{0, 1\}, \quad \forall r \in \mathcal{R} \tag{8}$$

The objective function (6) minimizes the total cost of the SB-VRP. Constraints (7) impose

that each customer is visited exactly once. Constraints (8) define the variables as binary. This model is solved using a commercial solver with a time limit $t_{postopt}$.

8 Experimental results

Our algorithm (denoted R&R) is implemented in C++ using Microsoft Visual C++ 2012. The tests are performed on an Intel Xeon CPU 2.8 GHz PC with 12 GB RAM using the Windows 7 operating system. CPU times are given in seconds. We use the callable library IBM ILOG CPLEX 12.6 with the default settings to solve the set partitioning problem.

8.1 Test Instances

We report computational tests on the set of 12 test instances proposed by the organizers of the Verolog Challenge 2014. The instances are classified in four sets: Small, Medium, Large1 and Large2 respectively characterized by 57, 206, 548 and 550 customers and 20, 41, 99 and 101 swap locations. The instances can have: only truck customers, only train customers or a mix. Table 2 summarizes the characteristics of the instances.

Set of instances	Small	Medium	Large1	Large2
No of instances	3	3	3	3
No of customers	57	206	548	550
No of swap locations	20	41	99	101

Table 2: Overview of the Benchmark Instances

8.2 Parameters setting

The procedure R&R uses a number of parameters. Table 3 summarizes their values.

Parameter	Value
n_{thread}	3
n_{init}	15
n_{final}	40
n_{elite}	2
n_{close}	4
$n_{closest}$	$0.5N$, $0.2N$, $0.1N$
p_{mut}	0.5
$t_{collaborative}$	600, 1200

Table 3: Parameters setting

R&R uses three threads ($n_{thread} = 3$). On each thread a population is initialized with 15 individuals and children are created until its size reaches 40. These values are determined following the guidelines given in Vidal *et al.* [20] and the values obtained in Cattaruzza *et al.* [1] while setting their procedure. Analogously the parameters n_{elite} and n_{close} are set to 2 and 4 respectively. These two parameters do not figure in this paper but are involved in the computation of the biased fitness presented in Section 5.2 (see Vidal *et al.* [20] for

details). The value $n_{closest}$ depends on the size of the particular instance that is solved: it is set to $0.5N$, $0.2N$ and $0.1N$ respectively for Small, Medium and Large instances. Preliminary extensive tests were made to determine these values. A new random generated chromosome is inserted in each population with a probability p_{mut} of 0.5. Finally, the organizers of the VeRoLog Challenge 2014 imposed a time limit of 600 seconds. Originally, the 600 seconds included the post-optimization phase. However, a fine tuning is required to determine the time to dedicate to R&R and to the post-optimization. Thus, we decided, when running our procedure to obtain the results presented here, to simply dedicate the whole 600 seconds to R&R and a maximum of 300 seconds for the set partitioning phase. The impact of the post-optimization phase is evaluated separately. A second value of $t_{collaborative}$, i.e., 1200 seconds is considered for comparison purposes. Again, a maximum of 300 seconds is allocated to the post-optimization phase.

8.3 Computational results

In the following, we analyze the efficiency of the R&R method, the impact of the parameters (number of used threads) and the impact of each phase on the quality of the obtained solutions. The algorithm is executed ten times. For each instance, the best solution is extracted after 600 and 1200 seconds. In order to solve the set partitioning problem, a time limit of 300 seconds is given to CPLEX after the end of the solution method.

Tables 4 reports best (column *best*) and average (column *avg*) solutions of R&R after respectively 600 seconds and 1200 seconds. We can notice that the average gap between average values and best values is relatively low. It is in average around 0.12% for small instances, 0.48% for medium instance and 1.14% for large instances. We can also notice that the percentage of improvement in best solution quality while increasing CPU time from 600 sec to 1200 sec is between 0.00% (for small instances) and 1.73% (for large instances). The average improvement between the time limits of 600 seconds and 1200 seconds is between 0.1% and 1%.

	Time limit R&R	600 seconds		1200 seconds	
Class	Customer type	avg	best	avg	best
Small	mix	4808.18	4802.38	4804.76	4802.38
	only trains	4724.81	4716.58	4721.96	4716.58
	only trucks	5007.74	4992.44	5001.30	4981.70
Medium	mix	7876.66	7819.35	7866.53	7810.93
	only trains	7827.93	7765.47	7815.83	7763.13
	only trucks	8112.88	8065.10	8102.58	8058.89
Large1	mix	21365.48	20983.90	21138.12	20760.30
	only trains	21026.51	20630.30	20877.81	20495.70
	only trucks	22026.64	21560.80	21987.20	21580.60
Large2	mix	25988.46	25844.20	25703.02	25529.50
	only trains	25670.90	25453.80	25365.26	25021.70
	only trucks	26253.76	26071.00	26205.02	25975.50

Table 4: Detailed solutions

Table 5 compiles best known solutions and compares the different methods proposed in

the literature. Column HG [7] gives the average results obtained by Huber and Geiger [7] with 30 runs of the so-called *all pre-selection* variant of their algorithm, on a Intel Xeon X5650 2.66 GHz. Column LCWGW [11] provides the results obtained by Lum et al. [11] on a 2012 MacBook Air. Column MCMMPZ [13] reports the average results by Miranda-Bront *et al.* [13] with 10 runs on a Intel Core i5-3320M. Our average results on 10 runs are recalled in column R&R. The computing time given to all these methods is 600 seconds. Recall that our method uses a post-optimization phase with a maximum of 300 seconds of additional computing time. Column BKS compiles the best known solutions among all solution values reported in the different papers. All best known values come from the different variants of the algorithm presented in Huber and Geiger [7], except for the small only-trains instance whose best known value comes from Table 4. The last line of the table indicates the overall gap of the methods against the best known solutions (summing over all instances). These results show that, if the method from Huber and Geiger [7] clearly outperforms the others, our R&R algorithm ranks rather well among the existing methods.

Class	Customer type	BKS	HG [7]	LCWGW [11]	MCMMPZ [13]	R&R
Small	mix	4797.85	4805.32	4959.00	4818.15	4808.18
	only trains	4716.58	4730.92	4873.05	4730.63	4724.81
	only trucks	4839.64	4860.06	5356.36	4913.09	5007.74
Medium	mix	7814.17	7818.87	8297.25	8040.17	7876.66
	only trains	7749.42	7885.75	8335.55	7957.60	7827.93
	only trucks	8021.88	8119.43	8628.37	8263.76	8112.88
Large1	mix	20471.40	20764.05	22051.40	21050.29	21365.48
	only trains	20215.26	20482.65	21317.00	20932.23	21026.51
	only trucks	21176.49	21457.48	22419.40	21782.45	22026.64
Large2	mix	25376.41	25617.77	26712.40	26291.65	25988.46
	only trains	24939.83	25331.66	26658.10	26013.09	25670.90
	only trucks	25835.85	26166.99	26712.40	26762.80	26253.76
Gap to BKS			1.19%	5.89%	3.18%	2.69%

Table 5: Best known solutions and algorithm comparison

Table 6 provides the gap between the second phase of the algorithm (repair procedure) and the first phase of the algorithm that provides an H-VRP solution which is not necessarily feasible. The table shows also the improvement provided by solving the set partitioning problem. Note that the first phase of the algorithm does not necessarily provide a lower bound to the problem. In fact, the H-VRP can obtain a route with only train customers. The cost of such a route can be improved by the correction procedure by visiting swap locations that can help reducing the cost of using a full train.

		Impact of the 2nd phase		Impact of set partitioning	
Class	Customer type	avg gap	max gap	avg gap	max gap
Small	mix	1.33%	2.96%	-0.10%	-0.54%
	only trains	0.00%	0.00%	-0.16%	-0.31%
	only trucks	2.81%	4.67%	-1.34%	-2.54%
Medium	mix	0.47%	1.33%	-1.12%	-1.78%
	only trains	0.00%	0.00%	-1.00%	-1.87%
	only trucks	2.35%	4.41%	-1.41%	-2.28%
Large1	mix	0.22%	0.93%	-0.76%	-2.03%
	only trains	0.00%	0.00%	-0.47%	-2.28%
	only trucks	0.68%	2.74%	-1.74%	-4.19%
Large2	mix	0.08%	0.56%	-0.81%	0.00%
	only trains	0.00%	0.00%	-1.01%	-0.24%
	only trucks	0.07%	1.03%	-0.67%	0.00%

Table 6: Impacts of the different phases

We can notice that the average deterioration of the objective function when repairing the solutions of the H-VRP in order to obtain feasible solutions for the SB-VRP is relatively small in average (0.75%). For example, for instances with only train customers, the average and maximum gaps are null. In this particular case, the routes obtained by the first phase of the algorithm are feasible and the solution would not need to be repaired. On the other side, this value tells the procedure never found beneficial to visit swap-location and perform (part of) a route with a truck, when it is assigned to a train (we recall that routing a truck costs less than routing a train). This gap increases to 1.24% in average for mix instances. For instances with only truck customers, this gap is higher in average (2.75%) and can reach 4.47%. We can also notice that average improvements when using the set partitioning procedure are higher for instances with only truck customers. This improvement can reach 4.19% for large instances.

Table 7 shows the CPU time (in seconds) spent by the set partitioning phase for each class. It shows that the time spent to solve the set partitioning problem is very low for small instances. On the other hand, this phase can use all the allocated time (300 seconds) for larger instances.

Time limit R&R	600 seconds	1200 seconds
Class	average CPU Time	average CPU Time
Small	19	35
Medium	251	277
Large1	300	300
Large2	300	300

Table 7: CPU Time of the set partitioning phase

Table 8 reports the impact of executing the algorithm on three threads rather that one thread. The columns *max* and *avg* report respectively the maximum and the average percentage of improvement obtained using three threads rather than only one thread. From Table 8 we can see that the impact of using several threads is higher for large instances. It can reach

2.79% for large mix instances. This improvement is reduced to 0.24% for small instances with only train customers.

Class	Customer type	max	avg
Small	mix	0.32%	0.10%
	only trains	0.24%	0.02%
	only trucks	0.56%	0.21%
Medium	mix	0.96%	0.23%
	only trains	1.04%	0.25%
	only trucks	0.95%	0.50%
Large1	mix	2.32%	1.30%
	only trains	2.57%	1.42%
	only trucks	2.13%	0.56%
Large2	mix	2.79%	1.50%
	only trains	1.99%	1.17%
	only trucks	2.29%	0.75%

Table 8: Impacts of using three threads

Table 9 reports the maximum and the average improvements obtained using three threads and the set partitioning phase rather than only one thread and no set partitioning phase. The columns *max* and *avg* report respectively the maximum and the average percentage of improvement. We can notice that this improvement can be relatively high for instances with only truck customers. This improvement is higher for large instances. From Tables 6, 8 and 9, we can conclude that using several threads and the set partitioning phase can help obtaining better results. In addition, these two steps do not require a heavy programming effort.

Class	Customer type	max	avg
Small	mix	0.68%	0.24%
	only trains	0.31%	0.20%
	only trucks	2.94%	1.78%
Medium	mix	2.13%	1.35%
	only trains	1.97%	1.21%
	only trucks	2.59%	1.90%
Large1	mix	2.93%	1.86%
	only trains	2.81%	1.66%
	only trucks	4.53%	2.34%
Large2	mix	2.79%	2.05%
	only trains	2.19%	1.68%
	only trucks	2.29%	1.10%

Table 9: Impacts of using three threads and set partitioning

8.4 New instances

We conducted new computations on a set of 20 new instances generated based on a subset of initial instances. These new instances are classified into 5 classes. Each class extends the four

initial mix instances. In the first class, the time horizon is multiplied by two. In the second class the time horizon is divided by two. All customers that cannot be served during this time horizon by a round trip are removed. In the third class of instances demand is divided by two (non-integer demands are rounded to the closest higher integer). The fourth class extends the four original instances by multiplying the demand by two. Clients with a demand higher than the capacity of the swap-body that cannot be visited using a train are removed. Clients with a demand higher than the capacity of two swap-bodies are also removed. In the fifth class of instances, the time horizon and the demand of each customer are divided by two (non-integer demands are rounded to the closest higher integer). All customers that cannot be served by a round trip during this time horizon are removed.

For each instance, the best solution is extracted after 600 seconds. In order to solve the set partitioning problem, a time limit of 300 seconds is given to CPLEX after the end of the solution method. The algorithm is executed five times. Tables 10 reports best (column *best*) and average (column *avg*) solutions of R&R after 600 seconds. #customers reports the number of customers for each instance.

Class	Size	#clients	best	avg
Class 1	Small	57	4292.07	4302.80
	Medium	206	7411.30	7532.78
	Large1	548	20008.50	20246.73
	Large2	550	21084.30	21328.52
Class 2	Small	39	3437.63	3444.68
	Medium	199	14450.50	14457.48
	Large1	520	37954.90	38031.15
	Large2	481	50052.80	50076.43
Class 3	Small	57	3975.87	3975.87
	Medium	206	6809.87	6820.14
	Large1	548	17606.80	17776.52
	Large2	550	22806.50	22874.95
Class 4	Small	56	6950.81	6981.30
	Medium	203	11015.70	11072.40
	Large1	548	30183.10	30409.00
	Large2	550	33601.20	33835.42
Class 5	Small	39	3222.84	3222.84
	Medium	199	14329.80	14335.62
	Large1	520	37338.10	37410.13
	Large2	481	50054.60	50066.77

Table 10: Detailed solutions for new instances

Through Table 10 we can notice that the average gap between average values and best values is relatively low. It is in average around 0.15% for small instances, 0.48% for medium instance and 0.55% for large instances. These small gaps show that our method is relatively robust. In this sense, remarkable is the fact that for the small instance of classes 3 and 5, the algorithm always finds the same best known solution.

9 Conclusions and perspectives

In this paper we addressed the Swap-Body Vehicle Routing Problem (SB-VRP), a variant of the Truck and Trailer Routing Problem. We proposed a solution scheme that we called Relax-and-Repair. This approach consists in solving a relaxed version of the SB-VRP and deriving a feasible solution by repairing the relaxed one. We embedded this approach within a population-based heuristic (memetic algorithm). During computation we stored all feasible routes in order to derive better solutions by solving a set-partitioning problem. To take advantages of nowadays multi-core machines, the algorithm is designed as a collaborative parallel population-based heuristic. Experimental results showed that the relax-and-repair algorithm is very competitive and allowed analyzing the impact of each phase on the quality of the obtained solutions. For example, the set-partitioning phase can improve the best obtained solution by more than 4%, and the the use of a three-cores machine can improve the solution by almost 3%.

The proposed relax-and-repair scheme can be adapted to solve complex industrial routing problems. In fact, the recent advances in the vehicle routing context showed that a large variety of rich vehicle routing problems can be solved efficiently. Relaxing complex constraints can allow to deal with efficiently solved rich vehicle routing problems. The repair procedure should be developed to obtain feasible solutions. The other parts of the algorithm can remain unchanged.

References

[1] Diego Cattaruzza, Nabil Absi, Dominique Feillet and Thibaut Vidal. A memetic algorithm for the Multi Trip Vehicle Routing Problem European Journal of Operational Research, 236(6):833–848, 2014.

[2] I-Ming Chao. A tabu search method for the truck and trailer routing problem. *Computers & Operations Research*, 29(1):33–51, 2002.

[3] Jean-François Cordeau, Gilbert Laporte, and Anne Mercier. A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational Research Society*, 52:928–936, 2001.

[4] Michael Drexl. Applications of the vehicle routing problem with trailers and transshipments. *European Journal of Operational Research*, 2012.

[5] Christophe Duhamel, Christophe Gouinaud, Philippe Lacomme, and Caroline Prodhon. A multi-thread graspxels for the heterogeneous capacitated vehicle routing problem. In *Hybrid Metaheuristics - Studies in Computational Intelligence*, volume 434, pages 237–269. Springer, 2013.

[6] Johanna C Gerdessen. Vehicle routing problem with trailers. *European Journal of Operational Research*, 93(1):135–147, 1996.

[7] Sandra Huber and Martin Josef Geiger. Swap body vehicle routing problem: A heuristic solution approach. In *Computational Logistics*, pages 16–30. Springer, 2014.

[8] Stefan Irnich. A unified modeling and solution framework for vehicle routing and local search-based metaheuristics. *INFORMS Journal on Computing*, 20(2):270–287, 2008.

[9] Shih-Wei Lin, Vincent F Yu, and Shuo-Yan Chou. Solving the truck and trailer routing problem based on a simulated annealing heuristic. *Computers & Operations Research*, 36(5):1683–1692, 2009.

[10] Shih-Wei Lin, Vincent F Yu, and Shuo-Yan Chou. A note on the truck and trailer routing problem. *Expert Systems with Applications*, 37(1):899–903, 2010.

[11] Oliver Lum, Ping Chen, Xingyin Wang, Bruce Golden, Edward Wasil. A Heuristic Approach for the Swap-Body Vehicle Routing Problem. In 14th INFORMS Computing Society Conference, pages 172-187, 2015.

[12] Jorge E Mendoza and Juan G Villegas. A multi-space sampling heuristic for the vehicle routing problem with stochastic demands. *Optimization Letters*, 7(7):1503–1516, 2013.

[13] Juan-José Miranda-Bront, Brian Curcio, Isabel Méndez-Díaz, Augustín Montero, Federico Pousa, and Paula Zabala. A cluster-first route-second approach for the Swap Body Vehicle Routing Problem. 2015.

[14] Christian Prins. A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research*, 31(12):1985–2002, 2004.

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

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

[17] Frédéric Semet and Eric Taillard. Solving real-life vehicle routing problems efficiently using tabu search. *Annals of Operations research*, 41(4):469–488, 1993.

[18] Paolo Toth and Daniele Vigo. The granular tabu search and its application to the vehicle routing problem. *INFORMS Journal of Computing*, 15(4):333–346, 2003.

[19] Puca Huachi Vaz Penna, Anand Subramanian, and Luiz Satoru Ochi. An iterated local search heuristic for the heterogeneous fleet vehicle routing problem. *Journal of Heuristics*, 19(2):201–232, 2013.

[20] Thibaut Vidal, Teodor Gabriel Crainic, Michel Gendreau, Nadia Lahrichi, and Walter Rei. A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. *Operations Research*, 60(3):611–624, 2012.

[21] Thibaut Vidal, Teodor Gabriel Crainic, Michel Gendreau, and Christian Prins. A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time windows. *Computers & Operations Research*, 40(1):475–489, 2013.

[22] Juan G Villegas, Christian Prins, Caroline Prodhon, Andrés L Medaglia, and Nubia Velasco. Grasp/vnd and multi-start evolutionary local search for the single truck and trailer routing problem with satellite depots. *Engineering Applications of Artificial Intelligence*, 23(5):780–794, 2010.

[23] Juan G Villegas, Christian Prins, Caroline Prodhon, Andrés L Medaglia, and Nubia Velasco. A grasp with evolutionary path relinking for the truck and trailer routing problem. *Computers & Operations Research*, 38(9):1319–1334, 2011.