Computers & Operations Research 78 (2017) 468–479

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

A general variable neighborhood search for the swap-body vehicle routing problem

Raca Todosijević [a,b,e,*], Saïd Hanafi [a], Dragan Urošević [b], Bassem Jarboui [c], Bernard Gendron [d]

[a] LAMIH UMR CNRS 8201 – Université de Valenciennes, 59313 Valenciennes Cedex 9, France
[b] Mathematical Institute, Serbian Academy of Sciences and Arts, Knez Mihailova 36, 11000 Belgrade, Serbia
[c] MODELIS, Faculté des Sciences Economiques et de Gestion de Sfax, Tunisie
[d] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT), Canada
[e] Inria Lille-Nord Europe, 40 Avenue Halley, 59650 Villeneuve d'Ascq, France

ARTICLE INFO

Available online 10 February 2016

Keywords:
Vehicle routing
Swap body
Variable neighborhood search

ABSTRACT

The Swap-Body Vehicle Routing Problem, a generalization of the well known Vehicle Routing Problem, can be stated as follows: the vehicle fleet consisting of trucks, semi-trailers, and swap bodies, is available at a single depot to serve a given set of customers. To serve a subset of customers, one may use either a truck carrying one swap body or a train (a truck with a semi-trailer attached to it) carrying two swap bodies. In both cases, a vehicle (a truck or a train) must perform a route starting and ending at the depot, so to satisfy demands of visited customers, maximal allowed route duration, allowed load on the used vehicle, and accessibility constraint of each customer. The accessibility constraint indicates whether a customer is allowed to be visited by a train or not. In addition, a set of swap locations is given where semi-trailers and swap bodies may be parked or swapped. The goal of the Swap-Body Vehicle Routing Problem is to minimize the total costs consisting of the fixed costs for using vehicles and costs for performing routes. In this paper, we propose two general variable neighborhood search heuristics to solve this problem. The quality of the proposed methods is evaluated on the instances provided by the organizers of VeRolog Solver Challenge 2014.

1. Introduction

The Swap-Body Vehicle Routing Problem (SBVRP) is stated as follows. Given a homogeneous fleet with an unlimited number of vehicles, located at the depot, which contains trucks, trailers and swap-bodies. Only swap bodies may be loaded but their maximum capacity Q must not be exceeded. From the available fleet, two vehicle combinations are allowed to be established: a truck carrying one swap-body (SB), or a truck and a semi-trailer (called *train*) carrying two swap-bodies. Using available vehicle combinations, one has to service a given set of customers so that each of them is visited exactly once while performing the route that starts and ends at the depot. Each customer i has a demand q_i that must be delivered, servicing time s_i needed to serve it as well as the accessibility constraint. Namely, some customers (called *truck customers*) can be visited only by using a truck while others (called *train customers*) may be visited either by using a truck or a train. However, even if a train departs from the depot carrying on two swap-bodies, it can visit truck customers after detaching semitrailer at specified locations called *swap-locations*. If a semi-trailer is detached at some swap-location from a truck, it must be reattached by the same truck before returning to the depot. Note that no exchange of swap-bodies between trucks is allowed on the route, i.e., a truck/train has to return to the depot with exactly the same swap-bodies it departed with. At the swap locations, it is allowed to perform the following actions:

- *Park action* – a truck parks a semi-trailer and continues the route with just one swap-body.
- *Pick Up action* – a truck picks up the semi-trailer with the swap-body, parked there at an earlier stage of the route.
- *Swap action* – a truck parks the swap-body, currently carried on, and picks up another swap-body from the semi-trailer, parked there at an earlier stage of the route.

* Corresponding author at: Mathematical Institute, Serbian Academy of Sciences and Arts, Knez Mihailova 36, 11000 Belgrade, Serbia.
E-mail addresses: racatodosijevic@gmail.com (R. Todosijević), said.hanafi@univ-valenciennes.fr (S. Hanafi), draganu@turing.mi.sanu.ac.rs (D. Urošević), bassem_jarboui@yahoo.fr (B. Jarboui), Bernard.Gendron@cirrelt.ca (B. Gendron).

http://dx.doi.org/10.1016/j.cor.2016.01.016
0305-0548/

- *Exchange action* – a truck parks a semi-trailer, exchanges the swap-bodies between a truck and a semi-trailer, and continues the route.

Note that it is not allowed to exchange load between swap-bodies, neither at swap locations nor at the customers sites, even if they belong to the same train. However, if a customer is visited by train, its demand may be loaded on two swap-bodies.

Each performed route r has three attributes: total distance, D_r^T, traveled by a truck in kilometers (km), total distance D_r^S, traveled by semi-trailer (if used) in km, and the total duration, τ_r, in hours. The route duration includes not only time needed to travel from one location to another, but also time spent serving customers on the route, as well as time spent performing actions (if any) at swap locations. However, the duration of each route is limited by the predefined time bound τ_{max}. The three aforementioned attributes together with the fixed cost for using a truck, C_F^T, and the fixed cost for using semi-trailer, C_F^S, induce the total cost of a route. More precisely, we denote C_D^T and C_D^S to be the cost per *km* traversed by a truck and a semi-trailer, respectively, and C_T to be cost for 1 h spent on a route. Then, the cost of a route r performed by a train is given as:

$$C_r = C_D^T \times D_r^T + C_D^S \times D_r^S + C_T \times \tau_r + C_F^T + C_F^S \tag{1}$$

Similarly, the cost of a route r performed by a truck is calculated as:

$$C_r = C_D^T \times D_r^T + C_T \times \tau_r + C_F^T \tag{2}$$

So, the objective of the SBVRP is to find a set of routes of minimum total cost so that in each route customer requirements, the allocated vehicle constraints, and maximal route duration constraints are respected.

To the best of our knowledge, there are just three heuristic approaches proposed for solving the SBVRP. Huber and Geiger [10] developed an Iterated Local Search (ILS) based heuristic that applies intra- and inter-tour local search on a random initial solution.

Lum et al. [15] proposed a three phase approach. The aim of the first two phases is to construct good initial solution using Simulated Annealing (SA) and postprocessing, while the third phase uses a Variable Neighborhood Descent (VND) to further improve the constructed solution. In Miranda-Bront et al. [16], the authors proposed a cluster-first, route-second approach which includes a GRASP and ILS metaheuristics.

The purpose of this paper is to contribute to the solution approaches for the SBVRP. In this paper we propose a mixed Integer programming (MIP) formulation of the SBVRP. However, due to its complexity and limitations of the current state-of-the-art MIP solvers, the MIP model cannot be used for solving even small test instances. Because of that and NP hardness of the SBVRP (since the VRP is a special case of the SBVRP), there is a need for a heuristic that is able to provide good quality solutions in a reasonable amount of time. So, in this paper we propose an efficient procedure for creating an initial solution of the SBVRP, as well as several neighborhood structures for SBVRP that are explored within heuristics based on Variable Neighborhood Search (VNS). One of the proposed heuristics executes tasks in the sequential way, while another executes four tasks at once using four available CPU cores. The quality of the proposed methods has been evaluated on the instances provided by the organizers of VeRolog Solver Challenge 2014.

The rest of the paper is organized as follows. In the next section, we give the literature overview regarding problems related to the SBVRP, while in Section 3, we provide a MIP model for the SBVRP. In Section 4, we describe a solution representation of the SBVRP and give steps of a procedure for creating an initial solution of the SBVRP. Section 5 contains description of main ingredients of the proposed VNS heuristics along with their pseudo-codes. Computational results obtained on the test instances are presented in Section 6, while Section 7 concludes the paper and give directions for future research.

2. Related problems

The truck and trailer routing problem (TTRP) is a variant of vehicle routing problem introduced in 2002 by Chao [2]. It is defined as follows. Given a limited-size fleet of trucks and trailers, and a set of customers to be serviced, where each truck/trailer has capacity constraint that must be respected, and each customer has demand that must be fulfilled using available fleet. However, not all customers are allowed to be visited by a truck pulling a trailer. Therefore, two types of customers are distinguished. The train customers that can be serviced by a truck pulling a trailer or by a truck alone, and the truck customers that can be serviced only by a truck. Truck can start a route from the depot with or without an attached trailer. If the trailer is attached to a truck, it must be uncoupled on the route and left at a vehicle customer before visiting a truck customer. If parking of trailer occurs, a truck performs sub-tour, routed at train customer where a trailer is parked, visits customers, picks up parked trailer, and continues its route. Note that it is not allowed to exchange trailers between trucks on the route, i.e., if a truck departs from the depot with the attached trailer, it must, also, return to the depot with the attached trailer. Also, note that transferring a load on a route between a truck and a trailer is allowed. The objective of the TTRP is, therefore, to find a set of the lowest cost vehicle routes that start and end at the depot so that each customer is serviced exactly once and the total demand of any vehicle route does not exceed the total capacity of the vehicles used in that route.

Applications of the TTRP arises in cases when customers may not be accessible with a truck–trailer combination due to some limitations. For example, some customers can be reached only via narrow roads and small bridges that cannot accommodate long vehicles, or sometimes using a truck–trailer combination is more time consuming than using just a truck to visit some customers. The real world applications of the TTRP are typical for dairy industry (see e.g [7,9])

Tabu search based heuristic for solving the TTRP was proposed by Chao [2] and Scheuerer [21]. Both proposed heuristics firstly build an initial feasible solution which is further improved by a tabu search. For building an initial solution, Chao [2] proposed a two phase procedure. In the first phase, the relaxed generalized assignment problem is solved to allocate customers to routes. After that, an insertion heuristic is used to construct routes. On the other hand, Scheuerer [21] used two cluster-first, route-second constructive heuristics, called *T-Sweep* and *T-Cluster*, to build an initial solution. Tabu search used in these two approaches explores neighborhood structures based on standard VRP moves as well as on the so-called *sub-tour root refining move* (consisted of attaching complete sub-tour to a different train customer).

In 2009, Lin et al. [13] proposed a heuristic approach that uses a simulated annealing to improve an initial solution. For constructing the initial solution, they used a procedure based on route-first, cluster-second approach. Villegas et al. [28] also used route-first, cluster-second based approach for constructing a solution. More precisely, they used route-first, cluster-second based approach within greedy randomized adaptive search procedure (GRASP), which is together with variable neighborhood search (VNS) and path relinking (PR) embedded in a hybrid method for solving the TTRP.

In 2010, Caramia and Guerriero [1] proposed an approach based on mathematical programming and local search for the TTRP. Mathematical programming is used to model and solve the customer-route assignment problem (CAP) (assigning customers to routes) and

the route-definition problem (RDP) (finding order of visiting customers in routes) one after another. Since the RDP may return infeasible solutions with disconnected sub-tours, the local search is used to retrieve a feasible solution.

In 2012, Derigs et al. [4] proposed heuristic approach for the TTRP that combines local search and large neighborhood search. The heuristic examines not only classic VRP neighborhoods but also TTRP-specific neighborhoods. Main advantage of the proposed heuristic is that it can be easily customized in order to tackle various variants of TTRP.

A hybrid approach that combines GRASP & iterated local search (ILS) with mathematical programming was proposed by Villegas et al. [29] in 2013. The constructive phase of GRASP & ILS is based on route-first, cluster-second approach, while the improvement phase is based on ILS. The routes in a solution returned by GRASP & ILS are used as columns in a set-partitioning formulation of the TTRP.

The TTRP has been extended to many variants: multi-depot, multi-period TTRP [20], the relaxed TTRP (RTTRP) where the limited-fleet constraint is dropped [14], the RTTRP with time windows [14], the TTRP without load transfer and the TTRP with time windows [4], the TTRP with time windows, site dependencies and heterogeneous fleet [23], etc. Drexl [5] introduced the vehicle routing problem with trailers and transshipment (VRPTT), an extension of the TTRP with time windows that includes vehicle-dependent routing cost, and existence of specific locations where the transfer of load between trucks and trailers is allowed. In addition, in the VRPTT, it is allowed to exchange trailers between trucks on the route, as well as, to transfer load from any truck to any trailer. From the VRPTT, the generalized truck and trailer routing problem (GTTRP) [6] is derived by imposing the fixed assignment of trailers to trucks.

The partial accessibility constrained vehicle routing problem (PACVRP) introduced by Semet [22] is a variant of TTRP in which each parking place for the trailer is root for only one sub-tour, and therefore, transfer of load between trucks and trailers is not allowed. Another variant of the TTRP is the vehicle routing problem with trailers (VRPT) presented by Gerdessen [7]. The differences of the VRPT with respect to the TTRP are: each customer can be visited either by a truck or a truck with a trailer; servicing time depends on a fact a customer is visited by truck pulling trailer or not; all customers have unit demand, and each trailer is parked exactly once.

The single truck and trailer routing problem with satellite depots (STTRPSD) studied by Villegas et al. [27] consists of satisfying the demand of a set of customers accessible only by a truck using a single truck with a detachable trailer. The trailer can be detached only in specified parking places called *satellite* depots. The authors proposed heuristic approach for solving the STTRPSD based on multi-start evolutionary local search, and a hybrid metaheuristic based on GRASP and variable neighborhood descent (VND).

3. Mixed integer programming formulation

In this section we propose an integer programming formulation for the SBVRP, based on the order of visiting customers. For that purpose, we use the following parameters:

- 0 : the depot
- C : the set of customers
- L : the set of swap locations
- N : the set of all nodes, i.e., $N = \{0\} + C + L$
- T : the set of trucks (unlimited number)
- S : the set of semi-trailers (unlimited number)
- q_i : the quantity that has to be delivered to the customer $i \in C$
- p_i : the boolean indicator of possibility to visit the customer $i \in C$ by a train, i.e., $p_i = 1$ if the customer i can be visited by train, otherwise $p_i = 0$
- s_i : service time at customer i
- O : set of operations, i.e., $O = \{P, U, W, E\}$, with $P =$ Park, $U =$ Pickup, $W =$ Swap and $E =$ Exchange
- ρ_l : time needed to perform action $l \in O$
- K : set of indices, hereafter also called 'moments'. Each index represent the ordering number of a node in a route
- Q : maximal capacity of a swap-body
- t_{ij}: time needed to traverse the arc (i,j)
- d_{ij}: distance between two nodes i and j

The following decision variables are used in our MIP formulation:

- $u_t^T = 1$ iff truck $t \in T$ is used
- $u_s^S = 1$ iff semi-trailer $s \in S$ is used
- $a_{s,t} = 1$ iff semi-trailer $s \in S$ is attached to truck $t \in T$
- $x_{i,k,t}^T = 1$ iff node $i \in N$ is visited as kth in the route performed by truck $t \in T$
- $x_{i,k,s}^S = 1$ iff node $i \in N$ is visited as kth in the route performed by semi-trailer $s \in S$
- $y_{i,j,t}^T = 1$ iff arc (i,j) traversed by truck $t \in T$
- $y_{i,j,s}^S = 1$ iff arc (i,j) traversed by semi-trailer $s \in S$
- $q_{k,t}^T$: capacity of truck $t \in T$ at moment k
- $q_{k,s}^S$: capacity of semi-trailer $s \in S$ at moment k

- $z^T_{i,k,t}$: amount delivered at 'moment' k at node i from truck $t \in T$
- $z^S_{i,k,s}$: amount delivered at 'moment' k at node i from swap-body $s \in S$
- $o^l_{i,k,s,t} = 1$ iff at kth node in route (i.e., $i \in L$) truck $t \in T$ and semi-trailer $s \in S$ perform operation $l \in O$

The proposed mixed integer programming model is stated as follows:

$$\min \sum_{t \in T} C^T_F \times u^T_t + \sum_{s \in S} C^S_F \times u^S_s + \sum_{i,j \in N, t \in T} C^T_D \times d_{ij} \times y^T_{i,j,t} + \sum_{i,j \in N, s \in S} C^S_D \times d_{ij} \times y^S_{i,j,s} + C_T \times \sum_{i,j \in N, t \in T} t_{ij} \times y^T_{i,j,t} + C_T \times \sum_{i \in C} s_i + C_T \times \sum_{i \in L, k \in K, s \in S, t \in T, l \in O} \rho_l \times o^l_{i,k,s,t} \quad (3)$$

Subject to:

$$u^S_s \le \sum_{t \in T} a_{s,t},\ s \in S \quad (4)$$

$$\sum_{s \in S} a_{s,t} \le u^T_t,\ t \in T \quad (5)$$

$$\sum_{i \in N} x^T_{i,k,t} \le u^T_t,\ t \in T,\ k \in K \quad (6)$$

$$\sum_{i \in N} x^S_{i,k,s} \le u^S_s,\ s \in S,\ k \in K \quad (7)$$

$$x^T_{0,0,t} = u^T_t,\ t \in T;\ x^S_{0,0,s} = u^S_s,\ s \in S \quad (8)$$

$$\sum_{k>0} x^T_{0,k,t} = u^T_t,\ t \in T; \sum_{k>0} x^S_{0,k,s} = u^S_s,\ s \in S \quad (9)$$

$$\sum_{i \in N} x^T_{i,k,t} = u^T_t - \sum_{1 \le h < k} x^T_{0,h,t} t \in T,\ k \in K \quad (10)$$

$$\sum_{i \in N} x^S_{i,k,s} = u^S_s - \sum_{1 \le h < k} x^S_{0,h,s} s \in S,\ k \in K \quad (11)$$

$$x^T_{i,k+1,t} + x^T_{i,k,t} \le 1,\ i \in N,\ t \in T,\ k \in K,\ k \le |K| - 1 \quad (12)$$

$$x^S_{i,k+1,s} + x^S_{i,k,s} \le 1,\ i \in N - L,\ s \in S,\ k \in K,\ k \le |K| - 1 \quad (13)$$

$$\sum_{k \in K, t \in T} x^T_{i,k,t} = 1,\ i \in C \quad (14)$$

$$y^S_{i,j,s} \le y^T_{i,j,t} + 1 - a_{s,t},\ i,j \in N,\ i \ne j,\ s \in S,\ t \in T \quad (15)$$

$$x^T_{i,k,t} \le x^S_{i,k,s} + 1 - a_{s,t}, \imath \in L,\ s \in S,\ t \in T,\ k \in K \quad (16)$$

$$x^T_{i,k,t} + a_{s,t} + x^S_{i,k-1,s} - 2 \le o^W_{i,k,s,t} + o^U_{i,k,s,t} \le 4 - x^T_{i,k,t} - a_{s,t} - x^S_{i,k-1,s},\ i \in L,\ s \in S,\ t \in T,\ k \in K,\ k \ge 1 \quad (17)$$

$$x^T_{i,k,t} + a_{s,t} - x^S_{i,k-1,s} - 1 \le o^P_{i,k,s,t} + o^E_{i,k,s,t} \le 3 - x^T_{i,k,t} - a_{s,t} + x^S_{i,k-1,s},\ i \in L, s \in S, t \in T, k \in K, k \ge 1 \quad (18)$$

$$o^P_{i,k,s,t} \le x^S_{i,k+1,s},\ i \in L,\ s \in S,\ t \in T,\ k \in K,\ k \le |K| - 1; \quad (19)$$

$$o^E_{i,k,s,t} \le x^S_{i,k+1,s},\ i \in L,\ s \in S,\ t \in T, k \in K, k \le |K| - 1; \quad (20)$$

$$o^W_{i,k,s,t} \le x^S_{i,k+1,s},\ i \in L,\ s \in S,\ t \in T, k \in K, k \le |K| - 1; \quad (21)$$

$$x^S_{i,k+1,s} \le 1 - o^U_{i,k,s,t},\ i \in L,\ s \in S,\ t \in T,\ k \in K, k \le |K| - 1; \quad (22)$$

$$q^S_{k,s} - Q(1 - o^E_{i,k,s,t}) \le q^T_{k+1,t} \le q^S_{k,s} + Q(1 - o^E_{i,k,s,t}),\ i \in L,\ s \in S,\ t \in T,\ k \in K,\ k \le |K| - 1; \quad (23)$$

$$q^T_{k,t} - Q(1 - o^E_{i,k,s,t}) \le q^S_{k+1,s} \le q^T_{k,t} + Q(1 - o^E_{i,k,s,t}),\ i \in L,\ s \in S,\ t \in T,\ k \in K,\ k \le |K| - 1; \quad (24)$$

$$q^S_{k,s} - Q(1-o^W_{i,k,s,t}) \le q^T_{k+1,t} \le q^S_{k,s} + Q(1-o^W_{i,k,s,t}),\ i \in L,\ s \in S,\ t \in T,\ k \in K,\ k \le |K|-1; \tag{25}$$

$$q^T_{k,t} - Q(1-o^W_{i,k,s,t}) \le q^S_{k+1,s} \le q^T_{k,t} + Q(1-o^W_{i,k,s,t}),\ i \in L,\ s \in S,\ t \in T,\ k \in K,\ k \le |K|-1; \tag{26}$$

$$0 \le z^T_{i,k,t} \le q_i x^T_{i,k,t},\ i \in C,\ k \in K,\ t \in T; \tag{27}$$

$$0 \le z^S_{i,k,s} \le q_i x^S_{i,k,s},\ i \in C,\ k \in K,\ s \in S; \tag{28}$$

$$\sum_{k \in K, t \in T, s \in S} (z^T_{i,k,t} + z^S_{i,k,s}) = q_i,\ i \in C; \tag{29}$$

$$q^T_{0,t} \le Q \times u^T_t,\ t \in T; \tag{30}$$

$$q^S_{0,s} \le Q \times u^S_s,\ s \in S; \tag{31}$$

$$q^T_{k,t} - \sum_{i \in C} z^T_{i,k,t} - Q \sum_{i \in L, s \in S} (o^W_{i,k,s,t} + o^E_{i,k,s,t}) \le q^T_{k+1,t} \le q^T_{k,t} - \sum_{i \in C} z^T_{i,k,t} + Q \sum_{i \in L, s \in S} (o^W_{i,k,s,t} + o^E_{i,k,s,t}), k \in K, , k \le |K|-1, t \in T; \tag{32}$$

$$q^S_{k,s} - \sum_{i \in C} z^T_{i,k,s} - Q \sum_{i \in L, t \in T} (o^W_{i,k,s,t} + o^E_{i,k,s,t}) \le q^S_{k+1,t} \le q^S_{k,s} - \sum_{i \in C} z^T_{i,k,s} - Q \sum_{i \in L, t \in T} (o^W_{i,k,s,t} + o^E_{i,k,s,t}), k \in K, k \le |K|-1, s \in S; \tag{33}$$

$$x^S_{i,k,s} \le p_i,\ i \in C,\ k \in K,\ s \in S; \tag{34}$$

$$x^T_{i,k,t} + x^T_{j,k+1,t} - 1 \le y^T_{i,j,t},\ i,j \in N,\ t \in T,\ k \in K,\ k \le |K|-1; \tag{35}$$

$$x^S_{i,k,s} + x^S_{j,k+1,t} - 1 \le y^S_{i,j,s},\ i,j \in N,\ s \in S,\ k \in K, k \le |K|-1; \tag{36}$$

$$\sum_{i,j \in N} t_{ij} \times y^T_{i,j,t} + \sum_{i \in C, k \in k} s_i \times x^T_{i,k,t} + \sum_{i \in L, k \in K, s \in S, l \in O} \rho_l \times o^l_{i,k,s,t} \le \tau_{max},\ t \in T \tag{37}$$

$$u^T_t, u^S_s, a_{s,t}, \in \{0,1\},\ s \in S,\ t \in T; \tag{38}$$

$$x^T_{i,k,t}, x^S_{i,k,s}, y^T_{i,j,t}, y^S_{i,j,s}, o^l_{i,k,s,t} \in \{0,1\},\ i,j \in N,\ k \in K,\ s \in S,\ t \in T,\ l \in O; \tag{39}$$

$$q^T_{k,t}, q^S_{k,s}, z^T_{i,k,t}, z^S_{i,k,s} \ge 0,\ i \in N,\ k \in K,\ t \in T,\ s \in S; \tag{40}$$

The objective function (3) minimizes the total cost. The first two terms represent the cost for using trucks and semi-trailers, respectively. The next two terms represent the costs induced by distance traveled by trucks and semitrailers, respectively. The last three terms represent costs induced by total traveling time, servicing time, and total time spent performing swap actions, respectively. Explanations of the constraints are as follows:

- Constraints (4) ensure that any semi-trailer $s \in S$ (if used) must be associated to some truck.
- Constraints (5) guarantee that any truck $t \in T$ (if used) can attach at most one semi-trailer.
- Constraints (6) (resp. (7)) impose that at each moment k for each truck (resp. semi-trailer) (if used) at most one node is visited.
- Constraints (8) imply that if a truck (a semi-trailer) is used, it must start a tour from the depot.
- Constraints (9) ensure that if a truck (a semi-trailer) is used, it comes back to the depot at some 'moment'.
- Constraints (10) (resp. (11)) imply that if a truck (resp. a semi-trailer) (if used) visits depot at some moment h before k, it cannot visit any node after this moment h.
- Constraints (12) and (13) mean that a truck and a semi-trailer, respectively cannot stay at any node expect that a semi-trailer can stay at swap locations.
- Constraints (14) guarantee that each customer is visited exactly once.
- Constraints (15) ensure that a semi-trailer cannot traverse any arc without the truck to which it is attached.
- Constraints (16) does no allow that a truck visits swap-location i in the kth 'moment' if a semi-trailer is not attached to it also (or it is already there).
- Constraints (17) ensure that if the track visits swap location i at kth moment, and the semi-trailer associated to it is located at that swap-location, then swap or pickup operation must be performed, otherwise this constraint is redundant. Similarly, constraints (18) ensure that if a track visits swap location i at kth moment with the semi-trailer attached to it, then park or exchange operation must be performed, otherwise this constraint is redundant.
- Constraints (19)–(21) imply that if the semi-trailer s is parked at moment k at swap-location i it must stay there in the next moment $k+1$, too.
- Constraints (22) mean that if pickup operation is performed at some moment, then a semi-trailer must leave swap-location where it was parked.
- Constraints (23)–(26) imply that if exchange or swap operation is performed, then loads between the truck and the semi-trailer are exchanged.
- Constraints (27)–(29) concern respecting demand of customers.
- Constraints (30)–(31) assure that the starting load of swap-bodies do not exceed the imposed capacity.

- Constraints (32) and (33) compute the load of swap-bodies after visiting customer at moment k. In case that no customers are visited and the exchange or swap operation is performed, those constraints are redundant.
- Constraints (34) ensure that truck does not a visit truck–customer with a semi-trailer attached to it if it is not allowed to do that.
- Constraints (35) (resp. (36)) indicate whether arc (i,j) is traversed or not with a truck (resp. a semi-trailer).
- Constraints (37) impose upper-bound on a tour duration.

The proposed MIP model has been implemented in CPLEX in order to verify its correctness and to solve some problem instances. However, because of NP hardness of the problem and insufficient memory to accommodate large size instances on the used computer (see Section 6 for the computer characteristics), CPLEX 12.6 mip solver succeed to solve just trivial instances with up to 10 nodes.

4. Initial solution

4.1. Solution representation

We represent a solution of the SBVRP as a set of routes. Each route is composed of a main tour and a set of sub-tours (if any). A main tour starts and ends at the depot, while a sub-tour starts and ends at some swap-location visited in a main tour (see Fig. 1). If a set of sub-tours is non-empty, then each sub-tour is rooted at some swap-location. Further in that case, each sub-tour contains either truck or train customers, while main-tour contains only train customers (if any) and swap-locations. Existence of a sub-tour in a route means that a train visits a swap-location, performs some swap-action (park or exchange), serves set of customers (truck or train customers), returns to the last visited swap-locations, and again performs some swap-action (pick-up or swap). In the case, that the pick-up action occurs, a train continues traversing the main-tour, while otherwise, a truck makes one more sub-tour rooted at the same swap-location. On the other hand, in the case that the set of sub-tour is empty, we distinguish two types of main tours: a train main tour and a truck main tour. The train main tour is achieved by visiting train customers using a train vehicle. On the other hand, truck main tour arises when truck is used to serve only truck customers, or only train customers, or both truck and train customers.

4.2. Constructive heuristic for creating an initial solution

In order to construct an initial solution, we used cluster-first, route-second approach (see Algorithm 1). In the first step, truck customers are clustered with respect to the depot and the swap locations which are considered as centers of clusters. The proximity of each truck customer i to a center j, denoted δ_{ji}, is estimated as a cost that will occur if the truck customer i is serviced via the center j. In the case that the depot is considered as a center, the cost is estimated as the cost of the tour performed by a truck going from the depot to a truck customer i and returning to the depot, i.e.,

$$\delta_{0i} = C_F^T + C_D^T \times (d_{0i} + d_{i0}) + C_T \times (t_{0i} + t_{i0} + s_i). \tag{41}$$

On the other hand, if a truck customer is served via a swap-location, the cost is calculated as the cost of the route traversed by train vehicle going from the depot to a swap-location j (parking semi-trailer), visiting a truck customer i, coming back to the swap-location (picking up semi-trailer) and finally coming back to the depot, i.e.,

$$\delta_{ji} = C_F^T + C_F^S + C_D^T \times (d_{0j} + d_{ji} + d_{ij} + d_{j0}) + C_D^S \times (d_{0j} + d_{j0}) + C_T \times (t_{0j} + t_{ji} + t_{ij} + t_{j0} + s_i + \rho_P + \rho_U). \tag{42}$$

After clustering truck customers, the routing is performed in each cluster in order to create sub-tours, as well as to create truck main tours. In order to achieve this, in each obtained cluster, we solve a VRP using Clarke and Wright heuristic [3] assuming that we use only truck vehicles which capacity equals to Q, and setting cluster center to serve as a depot in the considered VRP. Each route constructed in a cluster whose center is a swap-location represents a sub-tour (in our representation), while each route constructed in the cluster whose center is the depot represents a truck main tour.

In the last step, main tours are constructed. Main tours are obtained as solutions of a VRP in which train customers and swap-locations are taken into account. To each swap-location a demand equal to the total demand on the sub-tour (constructed in previous step) originating in it is assigned. Note that in the case that there is more than one sub-tour rooted at some swap-location, we create copies of the swap-location and set demand of each copy to be equal to the total demand of one sub-tour rooted at the swap-location. The resulting VRP

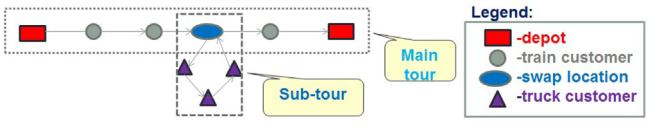

Fig. 1. Solution representation.

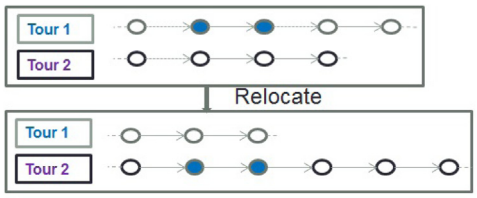

Fig. 2. Relocate move.

is again solved by Clarke and Wright heuristic, taking into account that we have the fleet of vehicles located at the depot which capacity is equal to 2*Q*. In addition, we imposed a constraint that the total demand of swap-location nodes included in one route must not be greater than *Q*. Under such constraint, the total demand of customers in the sub-tours will be loaded in a swap-body carried on a truck. The main purpose of this constraint is to ensure that the total demand of customers in the sub-tours is satisfied.

In Algorithm 1, the procedure `VRP (depot, customers, demands, capacity)` solves VRP defined with respect to a given `depot`, set of `customers`, set of `demands` of each customer and set of vehicles with a given `capacity`.

Algorithm 1. Procedure for creating an initial solution.

Algorithm 1: Procedure for creating an initial solution

Function `Initial Solution()`;
$\mathcal{C}_0 = \emptyset$; //cluster that corresponds to the depot
for $j = 1$ **to** $|L|$ **do**
  $\mathcal{C}_j = \emptyset$; //clusters that correspond to the swap-locations
end
for *each truck customer* i **do**
  calculate δ_{0i} using equation (41);
  for *each swap-location* $j \in L$ **do**
    calculate δ_{ji} using equation (42);
  end
end
//Cluster truck customers
for *each truck customer* i **do**
  $j^* \leftarrow \text{argmin}_{j \in \{0\} \cup L} \delta_{ji}$;
  $\mathcal{C}_{j^*} = \mathcal{C}_{j^*} \cup \{i\}$;
end
//Solve VRP in each cluster $\mathcal{C}_j$ to construct sub-tours and truck main tour
$S = \emptyset$;
for $j \in L \cup \{0\}$ **do**
  $S = S \cup \text{VRP}(j, \mathcal{C}_j, \{q_i : i \in \mathcal{C}_j\}, Q)$;
end
//Solve VRP w.r.t train customers and nodes obtained aggregating sub-tours (i.e., create train main tours)
$S = S \cup \text{VRP}(0, \{i : i \text{ train customer}\} \cup L, \{q_i : i \text{ train customer}\} \cup \{\sum_{k \in \mathcal{C}_j} q_k : j \in L\}, 2Q)$;
return S;

5. Variable neighborhood search

Variable neighborhood search (VNS) is a metaheuristic proposed in Mladenović and Hansen [17]. The basic variant of Variable neighborhood search (called Basic VNS) includes an improvement phase in which a local search is applied and one the so-called shaking phase used to hopefully resolve local minima traps. The local search and the shaking procedure, together with the neighborhood change step, are executed alternately until fulfilling a predefined stopping criterion. As the stopping criterion, most often, is used maximum CPU time allowed to be consumed by Basic VNS. From this basic VNS scheme, many variants of VNS have been derived (see e.g., [8] for recent survey) and successfully applied for solving many optimization problems (see e.g., [8,18,11,19,12,25]). However, the best known, as well as widely used VNS variant, is the so-called General VNS (GVNS). It uses more advanced local search procedure in the improvement phase than Basic VNS. The most common improvement procedures, used within GVNS, are based on variable neighborhood descent (VND) such as sequential VND [8], nested VND [25], cyclic VND [24], and so on.

5.1. Neighborhood structures

For a solution of the SBVRP, we define neighborhood structures adapting ones most commonly used in the VRP related problems. The moves that are used to constitute these neighborhoods may be divided into two groups: intra-route moves (that involve one main tour or sub-tour) and inter-route moves (that involve two tours (either main tours or sub-tours)). The intra route moves that we used are actually standard traveling salesman problem (TSP) moves:

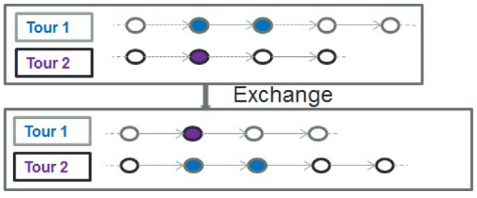

Fig. 3. Exchange move.

- 2-opt move – consists of inverting a part of a tour such that two edges of a given tour are broken down and two new edges are created;
- 1-opt move – is a special case of 2-opt move which includes inverting an edge of a given tour;
- OR-opt move Insertion neighborhood – is a special case of 3-opt move which involves relocation of one customer of a given tour somewhere else in the same tour.

Inter-route neighborhood moves are classified into two groups according to whether they include the relocation of a part of one tour to another (Fig. 2) or the exchange of two parts of two different tours (Fig. 3). These moves are further classified with respect to the tours involved in a move. Therefore, we distinguish the following eight moves: relocate a part from one main tour to another (`Relocate_main_to_main`); relocate a part from a main tour to a sub-tour (`Relocate_main_to_subtour`); relocate a part from a sub-tour to a main tour (`Relocate_subtour_to_main`); relocate a part from one sub-tour to another (`Relocate_subtour_to_subtour`); exchange parts of two main tour (`Exchange_main_main`); exchange parts of a main tour and a sub-tour (`Exchange_main_subtour`); exchange parts of two sub-tours (`Exchange_subtour_subtour`).

Note that in these moves some restrictions are imposed in order to preserve feasibility. Namely, we do not allow relocation of a part of a main tour which contains swap locations into a sub-tour. Similarly, a part of a sub-tour that contains truck customers cannot be relocated in a main tour if the resulting route would be served by a train. The similar restrictions are included in moves that exchange tour parts. It should be emphasized that relocate moves (`Relocate_subtour_to_main` and `Relocate_subtour_to_subtour`) also allow movement of whole sub-tours, and therefore sub-tour elimination. Similarly, `Exchange_subtour_subtour` move enables us to exchange two sub-tours within the same route or between two different routes. In order to avoid additional computation needed for feasibility checking, we treat as feasible moves only those whose execution lead to routes such that the total demand on their sub-tours is not greater than Q. In other words, our solution space contains only solutions such that if a route in a solution contains a swap location, at this swap location only park and pick up actions are executed.

The last neighborhood that we defined for SBVRP is one based on moves that involve replacement of one swap location by another. However, after some preliminary testing this neighborhood turns out to be non effective so, we decided not to explore it.

5.2. Local search and shaking

The previously described neighborhood structures have been exploited within the variable neighborhood search scheme. More precisely, the neighborhoods based on the similar move structures are embedded within the same variable neighborhood descent scheme. In that way we create three VNDs:

- VND_TSP – explores neighborhoods of a given solution S that are based on standard TSP moves. Its steps are presented in Algorithm 2.
- VND_relocate – uses the neighborhoods of a given solution S that involves relocation of a part of one tour to another in the way described in Algorithm 3.
- VND_exchange – explores neighborhoods of a given solution S obtained exchanging the parts of tours. The steps of VND_exchange are given in Algorithm 4.

Algorithm 2. VND_TSP.

Algorithm 2: VND_TSP

Function `VND_TSP` (S);
while *there is an improvement* **do**
  $S' \leftarrow$ `1-opt`(S);
  if (S' better than S) **then** $\{S \leftarrow S'$; **continue;**$\}$
  $S' \leftarrow$ `OR-opt`(S);
  if (S' better than S) **then** $\{S \leftarrow S'$; **continue;**$\}$
  $S' \leftarrow$ `2-opt`(S);
  if (S' better than S) **then** $S \leftarrow S'$;
end

Algorithm 3. VND_relocate.

Algorithm 3: VND_relocate

Function `VND_relocate` (S);
while *there is an improvement* **do**
  $S' \leftarrow$ `Relocate_subtour_to_main`(S);
  if (S' better than S) **then** $\{S \leftarrow S'$; **continue;**$\}$
  $S' \leftarrow$ `Relocate_main_to_main`(S);
  if (S' better than S) **then** $\{S \leftarrow S'$; **continue;**$\}$
  $S' \leftarrow$ `Relocate_main_to_subtour`(S);
  if (S' better than S) **then** $\{S \leftarrow S'$; **continue;**$\}$
  $S' \leftarrow$ `Relocate_subtour_to_subtour`(S);
  if (S' better than S) **then** $S \leftarrow S'$;
end

Algorithm 4. VND_exchange.

Algorithm 4: VND_exchange

Function `VND_exchange` (S) ;
while *there is an improvement* **do**
  $S' \leftarrow$ `Exchange_subtour_subtour`(S) ;
  if (S' better than S) **then** $\{S \leftarrow S'$; **continue;**$\}$
  $S' \leftarrow$ `Exchange_main_subtour`(S) ;
  if (S' better than S) **then** $\{S \leftarrow S'$; **continue;**$\}$
  $S' \leftarrow$ `Exchange_main_main`(S) ;
  if (S' better than S) **then** $S \leftarrow S'$;
end

Each of these VNDs uses the first improvement search strategy, i.e., as soon as a VND detects solution better than the current one, it resumes search starting from that solution. The presented VNDs are used as ingredients of a local search procedure used within the proposed GVNS. The proposed local search applies VND_TSP, VND_exchange and VND_relocate in that order, one after another.

Algorithm 5. Local search procedure.

Algorithm 5: Local search procedure

Function `LS` (S) ;
$S' \leftarrow$ `VND_TSP`(S) ;
if (S' better than S) **then** $S \leftarrow S'$;
$S' \leftarrow$ `VND_excange`(S) ;
if (S' better than S) **then** $S \leftarrow S'$;
$S' \leftarrow$ `VND_relocate`(S) ;
if (S' better than S) **then** $S \leftarrow S'$;

In order to resolve local optima traps, in which the proposed local search may be stuck, we propose the following Shaking procedure (Algorithm 6). The Shaking procedure has two parameters: solution S, and the number of iterations, k, to be performed within it. In each iteration, the procedure firstly choose a tour (main or sub-tour) at random, and then choose its part of random length such that a swap-location is not included in the chosen part. After that, each customer from the selected part is moved to another tour, keeping the feasibility and deteriorating the current objective value as low as possible. The obtained solution is set to be the new incumbent solution S, and the whole process is repeated until the maximum number of iterations (i.e., k) allowed is reached.

Algorithm 6. Shaking procedure.

Algorithm 6: Shaking procedure

Function `Shake` (S, k) ;
for $i = 1$ **to** k **do**
  Choose a tour (main or sub-tour) in S at random;
  Choose a part π of previously selected tour of random length;
  Relocate each customer in π to another tour in S;
end

5.3. General VNS – pseudo code

The proposed General VNS (see Algorithm 7) at an input requires the maximum CPU time allowed (parameter t_{max}) and the maximum number of iterations (parameter k_{max}) that may be performed in the shaking step. The procedure starts building an initial solution by using the procedure presented at Algorithm 1. After that, the generated solution is hopefully improved by applying alternately the shaking procedure (Algorithm 6) and the local search procedure (Algorithm 5) until the imposed CPU time limit is reached.

Algorithm 7. GVNS for solving the SB-VRP.

Algorithm 7: GVNS for solving the SB-VRP.

Function GVNS(k_{max}, t_{max})

$S \leftarrow$ Initial_Solution();
repeat
  $k \leftarrow 1$;
  while $k \leq k_{max}$ **do**
    $S' \leftarrow$ Shake(S, k) ;
    $S'' \leftarrow$ LS(S') ;
    $k \leftarrow k + 1$;
    if S'' *is better than* S **then**
      $S \leftarrow S''$; $k \leftarrow 1$;
    end
  end
until CpuTime() $> t_{max}$;
Return S

Besides the sequential GVNS (Algorithm 7), we implemented the parallel GVNS (see Algorithm 8). The parallel GVNS executes four tasks simultaneously. Each task, executed on one core, consists of applying the shaking procedure and the local search procedure, one after another. Note that since the shaking procedure generates a solution at random, the four shaking procedures, executed simultaneously, return four different solutions and therefore, solutions returned by the local search procedure applied on each of these solutions are not necessarily the same. The best solution obtained after executing these four tasks is examined to be set as the new incumbent solution. If the change of incumbent solution occurs, the search process is resumed starting from the new incumbent solution. The whole process is repeated until reaching the CPU time limit.

Algorithm 8. Parallel GVNS for solving VRP.

Algorithm 8: Parallel GVNS for solving VRP.

Function Paralel_GVNS(k_{max}, t_{max})

$S \leftarrow$ Initial_Solution();
repeat
  $k \leftarrow 1$;
  while $k \leq k_{max}$ **do**
    Execute four tasks simultaneously:
      Task 1:
        $S'_1 \leftarrow$ Shake(S, k) ;
        $S_1 \leftarrow$ LS(S'_1) ;
      Task 2:
        $S'_2 \leftarrow$ Shake(S, k) ;
        $S_2 \leftarrow$ LS(S'_2) ;
      Task 3:
        $S'_3 \leftarrow$ Shake(S, k) ;
        $S_3 \leftarrow$ LS(S'_3) ;
      Task 4:
        $S'_4 \leftarrow$ Shake(S, k) ;
        $S_4 \leftarrow$ LS(S'_4) ;
    $S' \leftarrow \text{argmin}\{f(S_1), f(S_2), f(S_3), f(S_4)\}$; // $f(S)$ denotes the value of a solution S
    $k \leftarrow k + 1$;
    if S'*is better than* S **then**
      $S \leftarrow S'$; $k \leftarrow 1$;
    end
  end
until CpuTime() $> t_{max}$;
Return S

6. Computational results

Both GVNS algorithms (Algorithms 7 and 8) have been coded in C language. Their performances have been disclosed on the benchmark instances provided by the organizers of VeRolog 2014 solver challenge (VeRoLog [26]). Three types of instances are distinguished:

Table 1
Computational results on benchmark instances.

Test instance	$\|L\|$	$\|C\|$	Huber and Geiger [10]		Lum et al. [15]	Miranda-Bront et al. [16]		Parallel GVNS		Sequential GVNS		dev
			Av. value	Best value	Best value	Av. value	Best value	Value	Time(s)	Value	Time(s)	(%)
small normal	20	57	4805.32	4804.97	4959.00	4818.15	**4797.55**	4847.63	66.19	4847.63	187.28	0.00
small all_with_trailer	20	57	4730.92	4730.92	4873.05	4730.63	**4730.63**	4731.02	6.08	4731.02	119.85	0.00
small all_without_trailer	20	57	4860.06	4839.64	5356.36	4913.09	**4855.62**	5249.18	1.96	5249.18	146.08	0.00
medium normal	41	206	7818.87	**7755.43**	8297.25	8040.17	7952.30	7834.78	497.37	7878.74	49.09	0.56
medium all_with_trailer	41	206	7885.75	7817.83	8335.57	7957.60	7847.30	7765.75	357.50	**7754.39**	594.58	−0.15
medium all_without_trailer	41	206	8119.43	**8045.47**	8628.37	8263.76	8221.32	8382.80	464.59	8398.80	594.20	0.19
large normal	99	548	20,764.05	20,524.54	22,051.40	21,050.29	20,889.20	**20,496.40**	599.60	20,570.20	591.81	0.36
large all_with_trailer	99	548	20,482.65	20,215.26	21,317.00	20,932.23	20,778.30	**20,066.40**	465.37	20,263.10	527.26	0.98
large all_without_trailer	99	548	21,457.48	**21,255.51**	22,419.40	21,782.45	21,683.60	22,310.60	430.16	22,477.60	599.22	0.75
preselection normal	101	550	25,617.77	**25,425.85**	26,712.40	26,291.65	26,146.00	25,443.20	583.27	25,647.00	596.15	0.80
preselection all_with_trailer	101	550	25,331.66	25,072.36	26,658.10	26,013.09	25,915.10	**24,965.10**	599.09	25,272.30	580.85	1.23
preselection all_without_trailer	101	550	26,166.99	**25,835.85**	26,712.40	26,762.80	26,524.50	26,515.90	573.17	26,631.70	550.12	0.44
Average			14,836.75	14,693.64	15,526.69	15,129.66	15,028.45	14,884.06	387.03	14,976.81	428.04	0.43

- normal – there are customers that can be visited by train, as well as those that must be visited only by a truck;
- all_without_trailer – all customers must be serviced only by a truck (the instances of this type are actually the VRP instances);
- all_with_trailer – all customers can be visited either by a truck or by a train.

All tests have been carried out on a computer with Intel i7-4900MQ CPU 2.80 GHz and 16 GB RAM. In all experiments parameters t_{max} and k_{max} have been set to the same values for both GVNS versions, sequential and parallel. After extensive testing k_{max} has been adjusted to 3, while t_{max} has been set to 600 s, as suggested by the organizers of VeRolog 2014 solver challenge.

The computational results have been presented in Table 1. The results of proposed GVNS heuristics are compared with those reported in Huber and Geiger [10], Lum et al. [15], and Miranda-Bront et al. [16]. In columns '$|L|$' and '$|C|$', we report the number of swap locations and the number of customers in the considered instance, respectively. Columns `Av. value` and `Best value` provide average and best results obtained in Huber and Geiger [10] and Miranda-Bront et al. [16]. Values found by Lum et al. [15] and the tested GVNS heuristics are reported in the corresponding Columns `Value`. In addition, for each GVNS version, we provide time in seconds (Column `Time`) spent to reach the solution whose value is reported in the table. Finally, in column `dev` we report the percentage deviation of value found by the sequential GVNS from the corresponding value provided by the parallel GVNS. The percentage deviation is calculated as:

$$\frac{\texttt{SequentialGVNS\_value} - \texttt{ParallelGVNS\_value}}{\texttt{ParallelGVNS\_value}} \cdot 100\%.$$

The best solution value for each test instance is boldfaced.

From the reported results, it follows that proposed GVNS heuristics are highly competitive comparing to the existing solution approaches for the SBVRP. In addition, they established new best known solution values for 4 (out of 12) test instances. Three of these new best known solution values are due to the parallel GVNS, while one is due to the sequential GVNS. Comparing the average solution values over the benchmark set, the compared approaches may be ranked as follows. The approach proposed by Huber and Geiger [10] is the best one, while the parallel GVNS and the sequential GVNS take the second and the third place, respectively, in the overall ranking. The approach described in Miranda-Bront et al. [16] is ranked as fourth, while the approach of Lum et al. [15] exhibits the worst performances. Moreover, the approach of Lum et al. does not hold any best known solution value. The advantage of the approach proposed by Huber and Geiger over our GVNS heuristics mainly comes from the instances "all_without_trailer", which may be seen as the pure VRP test instances. The reason for such behavior of our GVNSs stems from the fact that our GVNSs are rather focused on the SBVRP instances than on the VRP instances. However, it is interesting to note that if we exclude the instances "all_without_trailer" from the benchmark set, the parallel GVNS exhibits better performances than the ILS of Huber and Geiger. Namely, in this case the average of best values of ILS over the resulting benchmark set is 14,543.40, while the average value found by Parallel GVNS equals to 14,518.79.

From the results presented in Table 1, we may infer that the parallel GVNS is better option for solving the SBVRP than the sequential GVNS. Although the sequential GVNS offered slightly better solution than the parallel GVNS on `medium_all_with_trailer` instance on all the other instances parallel GVNS provided better solutions. Further, regarding average CPU time consumed by each of GVNS variants it follows that the parallel GVNS is about 40 s faster than the sequential GVNS, on the average. On small instances, where the parallel GVNS and the sequential GVNS end with the same solutions, the parallel GVNS is significantly faster than the sequential GVNS. However, there are also few instances where the sequential GVNS is faster than the parallel GVNS. Such outcome is expected in sense that the parallelization of GVNS does not necessarily imply reduction of time needed to reach final solution for the first time (except in the case both variants report the same solution as final), but implies speeding up the solution process. In general, the success of the parallel GVNS may be explained by the fact that it explores larger part of the solution space than the sequential GVNS within the same time limit. Namely, the parallel GVNS at each iteration generates four solutions on which local searches are applied, unlike the sequential GVNS which performs the local search on only one solution at each iteration.

7. Conclusions

In this paper we study the Swap Body Vehicle routing problem (SBVRP) which is a generalization of the Vehicle routing problem (VRP). The main differences of the SBVRP compared to the VRP are: a vehicle fleet constituting of trucks, semi-trailers and swap bodies; valid

vehicle combinations are a truck carrying one swap body and a train (a truck with a semi-trailer) carrying two swap bodies; the accessibility constraints, i.e., not all customers are allowed to be visited by train; a set of swap locations were semi-trailers and swap bodies may be parked or swapped.

To tackle this NP hard problem, we propose a mixed Integer programming (MIP) formulation for the SBVRP, a constructive heuristic for generating an initial solution, and two general variable neighborhood search (GVNS) heuristics. The proposed constructive heuristic is based on the cluster-first and the route-second approach, where the clustering is performed around the depot and swap locations, while the routing is performed using Clarke & Wright heuristic to solve VRPs in each cluster, as well as to determine tours that contain swap locations. Besides efficient neighborhood structures for the VRP, several problem specific neighborhood structures have been introduced. Based on these neighborhoods, three variable neighborhood descent heuristics are developed and integrated in the local search used within GVNSs. One of the proposed GVNS heuristics uses four available CPU cores to execute four tasks in parallel, while the other executes tasks one after another. Both proposed heuristics were tested on the instances provided by the organizers of VeRolog Solver Challenge 2014. The computational results reveal the superiority of the parallel GVNS (that executes four tasks in parallel) over the sequential GVNS (that executes tasks one after another). In addition, the proposed GVNS heuristics turned out to be highly competitive comparing to the previous solution approaches for the SBVRP.

The future work may include proposing exact methods for the SBVRP based on the column generation (e.g., Branch and price algorithm), as well as hybrid approaches that combine exact methods and one of the proposed sequential and parallel GVNS heuristics. Further, the future work may include development of GVNS based heuristics for problems related to the SBVRP.

Acknowledgments

This work was supported by the Centre National de la Recherche Scientifique (CNRS), by the Campus interdisciplinaire de recherche, d'innovation technologique et de formation Internationale sur la Sécurité et l'Intermodalité des Transports (CISIT), and by the Laboratoire d'Automatique, de Mécanique et d'Informatique industrielles et Humaines (LAMIH) of the Université de Valenciennes et du Hainaut-Cambrésis.

References

[1] Caramia M, Guerriero F. A heuristic approach for the truck and trailer routing problem. J Oper Res Soc 2010;61(7):1168–80.
[2] Chao I-MC. A tabu search method for the truck and trailer routing problem. Comput Oper Res 2002;29(1):33–51.
[3] Clarke G, Wright JW. Scheduling of vehicles from a central depot to a number of delivery points. Oper Res 1964;12(4):568–81.
[4] Derigs U, Pullmann M, Vogel U. Truck and trailer routing-problems, heuristics and computational experience. Comput Oper Res 2013;40(2):536–46.
[5] Drexl M. On some generalized routing problems [Ph.D. thesis]. Faculty of Business and Economics, RWTH Aachen University; 2007.
[6] Drexl M. Branch-and-price and heuristic column generation for the generalized truck-and-trailer routing problem. J Quant Methods Econ Bus Adm 2011;12:5–38.
[7] Gerdessen JC. Vehicle routing problem with trailers. Eur J Oper Res 1996;93(1):135–47.
[8] Hansen P, Mladenović N, Pérez JAM. Variable neighbourhood search: methods and applications. Ann Oper Res 2010;175(1):367–407.
[9] Hoff A. Heuristics for rich vehicle routing problems [Ph.D. thesis]. Molde University College; 2006.
[10] Huber S, Geiger MJ. Swap body vehicle routing problem: a heuristic solution approach. In: Computational logistics. Springer, Berlin; 2014. p. 16–30.
[11] Jarboui B, Derbel H, Hanafi S, Mladenović N. Variable neighborhood search for location routing. Comput Oper Res 2013;40(1):47–57.
[12] Lazić J, Todosijević R, Hanafi S, Mladenović N. Variable and single neighbourhood diving for mip feasibility. Yugosl J Oper Res 2014. http://dx.doi.org/10.2298/YJOR140417027L.
[13] Lin S-W, Yu VF, Chou S-Y. Solving the truck and trailer routing problem based on a simulated annealing heuristic. Comput Oper Res 2009;36(5):1683–92.
[14] Lin S-W, Yu VF, Chou S-Y. A note on the truck and trailer routing problem. Expert Syst Appl 2010;37(1):899–903.
[15] Lum O, Chen P, Wang X, Golden B, Wasil E. A heuristic approach for the swap-body vehicle routing problem. In: 14th INFORMS computing society conference; 2015. p. 172–87.
[16] Miranda-Bront JJ, Curcio B, Méndez-Dıaz I, Montero A, Pousa F, Zabala P. A cluster-first route-second approach for the swap body vehicle routing problem. Working paper. ⟨http://www.optimization-online.org⟩; 2015.
[17] Mladenović N, Hansen P. Variable neighborhood search. Comput Oper Res 1997;24(11):1097–100.
[18] Mladenović N, Todosijević R, Urosević D. An efficient general variable neighborhood search for large travelling salesman problem with time windows. Yugosl J Oper Res 2013;23(1):19–31.
[19] Mladenović N, Todosijević R, Urošević D. Two level general variable neighborhood search for attractive traveling salesman problem. Comput Oper Res 2014;52:341–8.
[20] Scheuerer S. Neue tabusuche-heuristiken fr die logistische tourenplanung bei restringierendem anhngereinsatz, mehreren depots und planungsperioden [Ph.D. thesis]. School of Business, University of Regensburg; 2004.
[21] Scheuerer S. A tabu search heuristic for the truck and trailer routing problem. Comput Oper Res 2006;33(4):894–909.
[22] Semet F. A two-phase algorithm for the partial accessibility constrained vehicle routing problem. Ann Oper Res 1995;61(1):45–65.
[23] Semet F, Taillard E. Solving real-life vehicle routing problems efficiently using tabu search. Ann Oper Res 1993;41(4):469–88.
[24] Todosijević R, Mjirda A, Mladenović M, Hanafi S, Gendron B. A general variable neighborhood search variants for the travelling salesman problem with draft limits. Optim Lett 2014. http://dx.doi.org/10.1007/s11590-014-0788-9.
[25] Todosijević R, Urošević D, Mladenović N, Hanafi S. A general variable neighborhood search for solving the uncapacitated r-allocation p-hub median problem. Optim Lett 2015. http://dx.doi.org/10.1007/s11590-015-0867-6.
[26] VeRoLog. Swap-body vehicle routing problem. ⟨http://verolog.deis.unibo.it/news-events/general-news/verolog-solver-challenge-2014⟩; 2014.
[27] Villegas JG, Prins C, Prodhon C, Medaglia AL, Velasco N. Grasp/vnd and multi-start evolutionary local search for the single truck and trailer routing problem with satellite depots. Eng Appl Artif Intell 2010;23(5):780–94.
[28] Villegas JG, Prins C, Prodhon C, Medaglia AL, Velasco N. A grasp with evolutionary path relinking for the truck and trailer routing problem. Comput Oper Res 2011;38(9):1319–34.
[29] Villegas JG, Prins C, Prodhon C, Medaglia AL, Velasco N. A matheuristic for the truck and trailer routing problem. Eur J Oper Res 2013;230(2):231–44.