WORKING PAPER SERIES

A hybrid algorithm for the vehicle routing problem with three-dimensional loading constraints and mixed backhauls

Henriette Koch/Maximilian Schlögell/Andreas Bortfeldt

Working Paper No. 11/2017

Impressum (§ 5 TMG)

Herausgeber:
Otto-von-Guericke-Universität Magdeburg
Fakultät für Wirtschaftswissenschaft
Der Dekan

Verantwortlich für diese Ausgabe:
H. Koch, M. Schlögell and A. Bortfeldt
Otto-von-Guericke-Universität Magdeburg
Fakultät für Wirtschaftswissenschaft
Postfach 4120
39016 Magdeburg
Germany

http://www.fww.ovgu.de/femm

Bezug über den Herausgeber
ISSN 1615-4274

A hybrid algorithm for the vehicle routing problem with three-dimensional loading constraints and mixed backhauls

Henriette Koch[*][†] Maximilian Schlögell[*] Andreas Bortfeldt[*]

Abstract

In this paper, a variant of the vehicle routing problem with mixed backhauls (VRPMB) is presented, i.e. goods have to be delivered from a central depot to linehaul customers, and, at the same time, goods have to be picked up from backhaul customers and brought to the depot. Both types of customers can be visited in mixed sequences.

The goods to be delivered or picked up are three-dimensional (cuboid) items. Hence, in addition to a routing plan, a feasible packing plan for each tour has to be provided considering a number of loading constraints. The resulting problem is the vehicle routing problem with three-dimensional loading constraints and mixed backhauls (3L-VRPMB).

The simultaneous transport of linehaul and backhaul items presents a particular challenge of the problem. We consider two different loading variants in order to avoid any reloading during the tour: (i) rear loading with separate linehaul and backhaul sections and (ii) loading at a long side.

In order to solve the problem, we propose a hybrid metaheuristic consisting of a reactive tabu search for the routing problem and different packing heuristics for the loading problem. Numerical experiments are reported with benchmark instances from the literature for the one-dimensional VRPMB to examine the performance of the routing algorithm and with newly generated instances for the 3L-VRPMB.

Keywords: vehicle routing, backhauls, tabu search, packing

[*]Department of Management Science, Otto-von-Guericke-University Magdeburg, 39106 Magdeburg, Germany

[†]Corresponding author

1 Introduction

In 2010, the average empty running rate – i.e. the share of trucks driving without transporting any goods – in the European Union amounted to 24 % (de Angelis, 2011). This occurs, for example, if vehicles return empty from their deliveries. By incorporating the pickup of goods (*backhauling*) during the tours into the logistics system, empty runs can be reduced which subsequently leads to a reduction in travelled distances, fuel consumption and CO_2 emission. Therefore, vehicle routing problems (VRPs) with backhauls also gain increasing attention in research.

While backhaul problems can be modelled in different variants (cf. e.g. Parragh et al., 2008; Irnich et al., 2014), this paper will be focused on the VRP with mixed backhauls (VRPMB). In this problem variant, goods either have to be delivered to customers (*linehaul*) or picked up from them (*backhaul*). The sequence of linehaul and backhaul customers within a tour can be chosen arbitrarily.

Moreover, we aim to provide a more realistic modelling of the transportation of (bulky) goods which are of a size that cannot be neglected to ensure feasibility when planning the tours. Therefore, the transported goods are assumed to be three-dimensional (3D) cuboid items. Each solution of the problem must, thus, be equipped with a feasible packing plan per route. A particular challenge of the problem is to transport linehaul and backhaul items simultaneously on the same vehicle. In order to avoid any reloading during a tour, two different loading approaches are considered: (i) loading from the rear side with horizontal separation of the loading space into a delivery section and a pickup section and (ii) loading from one long side. The side from which items are loaded and unloaded is subsequently called loading side. The resulting problem belongs to the group of VRPs with three-dimensional loading constraints (3L-VRPs) which was introduced by Gendreau et al. (2006).

We propose a hybrid algorithm for solving the three-dimensional VRPMB. The underlying routing problem is solved with a reactive tabu search (RTS) based on the approach of Nagy et al. (2013). In order to solve the packing subproblem, different packing heuristics have been implemented which can be chosen alternatively. They were tested and compared concerning their performance.

The remainder of this paper is organized as follows: A detailed problem description is presented in Section 2. In Section 3, an overview of the relevant literature is given. The proposed RTS and the packing heuristics are described in Section 4. In Section 5, the experiment set-up is described, and the results are presented and analysed. Finally, the paper concludes with a summary and an outlook to future research in Section 6.

2 Problem Description

Let $G = (N, E)$ be a weighted, directed graph with the node set $N = \{0, 1, \ldots, n\}$, where node 0 represents the depot and the nodes $1, \ldots, n$ represent the n customer locations, and the edge set $E = \{(i, j) : i, j \in N\}$. The customers are divided into l linehaul customers and b backhaul customers, i.e. $N = \{0, 1, \ldots, n\} = \{0, 1, \ldots, l, l+1, \ldots, l+b\}$. Furthermore, let c_{ij} be the cost corresponding to edge $(i, j) \in E$. A set $I_i = \{1, \ldots, m_i\}$ of m_i cuboid items (boxes) is assigned to each customer i ($i \in N \setminus \{0\}$) which must either be delivered to them (linehaul) or picked up from them (backhaul). Each item I_{ik} ($i \in N \setminus \{0\}, k \in I_i$) has a known length l_{ik}, width w_{ik}, height h_{ik} and weight d_{ik}, and is assigned with a fragility flag f_{ik} indicating whether it is fragile ($f_{ik} = 1$) or not ($f_{ik} = 0$). v_{max} identical vehicles are available with a given weight capacity D and a three-dimensional cuboid loading space of length L, width W and height H.

A solution for the problem must contain information about the allocation of customers to routes, the customer sequences of the routes and the corresponding packing plans.

A packing plan P contains placements for one or more items. It is feasible if it fulfils the following conditions: (P1) all items lie entirely within their loading space, (P2) any two items which are placed simultaneously in one loading space must not overlap, (P3) all items must be placed orthogonally to the loading space edges. Moreover, the following additional packing constraints must be adhered to (cf. Gendreau et al., 2006):

(PC1) **Fixed vertical orientation:** The items can be rotated by 90° on the horizontal plane, but the height dimensions are fixed.

(PC2) **Vertical stability:** Each item must be supported by a given percentage α by the top face of other items or the container floor.

(PC3) **Fragility:** A non-fragile item cannot be placed on top of a fragile item, whereas fragile items can be placed on top of any other item.

(PC4) **LIFO:** The items must be loaded and unloaded solely by straight movements towards the loading side. Therefore, it must be ensured that the (un-)loading is not blocked by items that are delivered later or have already been picked up.

Let l_1 and l_2 be two linehaul customers and b_1 and b_2 two backhaul customers. Assuming l_1 precedes l_2 in a given tour, no item of l_2 must be positioned between the loading side and any item of customer l_1 or above such item. Analogously, if b_1 precedes b_2 in a tour, no item of b_1 can be positioned between the loading side and any item of customer b_2 or above such item. Furthermore, if b_1 precedes l_1 in a tour, no item of b_1 may be placed between the loading side and any item of l_1 or above such item.

The LIFO policy implies that the reloading of any item during the tour is forbidden. Therefore, this constraint is particularly challenging considering that linehaul and backhaul items are transported simultaneously. Two alternative loading approaches are applied here in order to avoid any reloading effort.

In the first variant, double-decker vehicles are used. These vehicles are rear-loaded (the loading side is the rear side) and the loading space is separated horizontally so that two separate compartments are available for each type (linehaul or backhaul). This way, the LIFO constraint must not be considered w.r.t. a mixture of linehaul and backhaul items. It is assumed that both compartments are of the same size. In the following, this variant will be referred to as loading space partition (LSP).

Secondly, side loading (SL) is applied for which so-called tautliners are used. These vehicles can only be loaded and unloaded from the side (the loading side is one long side). An example is illustrated in Figure 1. By loading linehaul (light grey) and backhaul (dark grey) items from opposing sides (cabin and rear side), space is created for backhaul items when linehaul items are unloaded. L_{LH} and L_{BH} represent the loading lengths (i.e. maximum front edge) of all linehaul and backhaul items, respectively, which are currently in the loading space. In order to avoid any overlapping, the sum of both lengths must not exceed L. The LIFO constraint must be considered as above ensuring that the (un-)loading is not blocked. In the example in Figure 1, items 2 and 3 must not be delivered after item 1. Moreover, the constraint must also be considered along the length axis ensuring that the unloading of linehaul items successively creates space for backhaul items. Therefore, item 4 may not be delivered after item 1.

A feasible route R is a sequence of locations $(0, i_1, \ldots, i_{n_r}, 0)$ which fulfils the following conditions: (R1) it starts and ends at the depot, (R2) it comprises each customer $i \in$

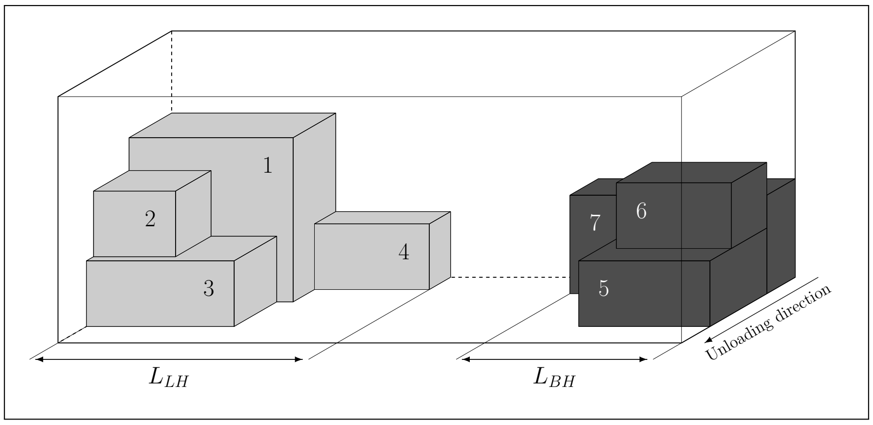


Figure 1: Side loading

$R \setminus \{0\}$ exactly once, (R3) the total weight of all items transported simultaneously does not exceed the vehicle weight capacity D, and (R4) a feasible packing plan P_L exists for all linehaul customers in R at the beginning of the tour and a feasible packing plan P_B exists for all backhaul customers at the end of the tour.

Let v be the number of used vehicles in a solution. Assuming each vehicle travels exactly one tour, a solution consists of a set of v triples $(R_t, P_{t,L}, P_{t,B})$ containing a route R_t for each vehicle t $(t = 1, \dots, v)$ and the corresponding packing plans $P_{t,L}$ and $P_{t,B}$. A solution is feasible if (S1) all routes R_t and packing plans $P_{t,L}, P_{t,B}$ $(t = 1, \dots, v)$ are feasible, (S2) each packing plan $P_{t,L}$ $(P_{t,B})$ contains all of the respective linehaul (backhaul) items (and no others) of all customers visited in R_t $(t = 1, \dots, v)$, (S3) each customer $i \in N \setminus \{0\}$ is assigned to exactly one route, (S4) the number of used vehicles v does not exceed the number of available vehicles v_{max}. Moreover, a feasible solution for the problem with SL approach must also adhere to the restriction (S5) that the linehaul and backhaul items that are transported *simultaneously* at any given moment in a route R_t $(t = 1, \dots, v)$ do not overlap, i.e. the sum of the lengths L_{LH} and L_{BH} must never exceed L (see above).

A feasible solution is to be found that minimizes the total travel distance (TTD). The problem can be classified as a 3L-VRP with mixed backhauls (3L-VRPMB).

3 Literature Review

The 3L-VRPMB has – to the best of our knowledge – not been considered in any scientific publications yet. Therefore, the following literature review is focused on the one-dimensional VRPMB and problem variants of the 3L-VRP. In addition, some relevant studies about the VRP with two-dimensional loading constraints are shortly mentioned.

3.1 Vehicle routing problem with mixed backhauls

The (one-dimensional) VRPMB has been studied intensively in the past decades. Straightforward heuristic solution approaches were primarily used in the beginning. They include, for example, savings and insertions heuristics (Golden et al., 1985; Casco et al., 1988), or cluster-first-route-second heuristics (Halse, 1992). In the recent past, the trend shifted towards the use of metaheuristics. One of the first metaheuristics for the VRPMB was presented by Wade and Salhi (2004) who suggested an ant colony optimization (ACO) approach. Crispim and Brandão (2005) presented a hybrid approach consisting of a tabu search (TS) and a variable neighbourhood descent. Ropke and Pisinger (2006) presented an adaptive large neighbourhood search (ALNS) for a great variety of VRPs with backhauls. More recently, Nagy et al. (2013) proposed a reactive tabu search (RTS) to solve the VRPMB which is the basis for the solution approach presented in this paper. In addition, they also considered a problem variant where a mixture of linehaul and backhaul customers in a tour is only allowed if a given percentage of the vehicle capacity is available. Hence, if the percentage equals 100 %, the VRP with clustered backhauls (VRPCB) is considered, i.e. all linehaul customers have to be visited before the backhaul customers within a tour. Further recent approaches include ACO (Wassan et al., 2013), adaptive local search (Avci and Topaloglu, 2015), or evolutionary algorithms (García-Nájera et al., 2015).

3.2 Vehicle routing problems with loading constraints

The capacitated VRP (CVRP) with three-dimensional loading constraints (3L-CVRP) was first presented by Gendreau et al. (2006), who also introduced the above mentioned constraints regarding the packing subproblem. Subsequently, the problem was studied by various researchers (e.g. Tarantilis et al., 2009; Fuellerer et al., 2010; Bortfeldt, 2012). Moura (2008) and Moura and Oliveira (2009) were the first to deal with the 3L-VRP with time windows. They consider two objective criteria. Namely, the TTD and the number of tours as common in research regarding VRPs with time windows (VRPTW). Furthermore, Moura (2008) also considered the maximization of the utilized volume as another optimization objective. Wei et al. (2014) address the 3L-CVRP with a heterogeneous vehicle fleet.

The routing problem is usually tackled with a metaheuristic approach, e.g., genetic algorithm (Moura, 2008; Miao et al., 2012), TS (Gendreau et al., 2006; Tarantilis et al., 2009; Wang et al., 2010; Ma et al., 2011; Wisniewski et al., 2011; Zhu et al., 2012; Tao and Wang, 2015), or ACO (Fuellerer et al., 2010). Since solving the packing problem requires comparatively much computing time as the packing procedure is called very frequently, the packing problem is often solved by applying simple construction heuristics, e.g. based on bottom-left and touching area heuristics. More complex packing approaches are, for example, applied by Bortfeldt (2012) (tree-search) or Zhang et al. (2015) (local-search based approach). Furthermore, Escobar-Falcón et al. (2016) used an exact approach to solve the routing subproblem and a GRASP algorithm to solve the packing problem for the obtained routes.

So far, the underlying VRP was mainly assumed to be a CVRP or VRPTW. Variants with pickup and delivery have not been studied intensely yet. Bortfeldt et al. (2015) approach the 3L-VRP with clustered backhauls (3L-VRPCB) using ALNS and variable neighbourhood search for the routing problem and a tree search procedure for the packing problem. Bartók and Imreh (2011) and Männel and Bortfeldt (2016) studied the pickup and delivery problem with three-dimensional loading constraints, i.e. goods are not transported between customer locations and a depot but from a loading location to an unloading location (that are not the depot). A detailed overview of the literature about the 3L-VRP is provided in Pollaris et al. (2015).

In addition, VRPs with backhauls have been studied considering two-dimensional (2D) loading constraints, e.g. Dominguez et al. (2015) (clustered backhauls), Pinto et al. (2015) (mixed backhauls) or Zachariadis et al. (2016) (simultaneous delivery and pickup). Both Pinto et al. (2015) and Zachariadis et al. (2016) consider variants with simultaneous transportation of linehaul and backhaul items and both do not allow rearrangements either. Pinto et al. (2015) approach it by first finding a feasible packing plan for linehaul items, and identifying free stripes where backhaul items can be placed. Zachariadis et al. (2016) utilize separated loading spaces.

4 Hybrid solution approach

Being a generalization of the CVRP, the 3L-VRPMB is also an NP-hard optimization problem (cf. e.g. Toth and Vigo, 2014). In order to find high-quality solutions within reasonable computing time, a metaheuristic framework is applied to solve the routing problem. As in many previous works, the packing problem is tackled with construction heuristics. In the following subsections, both parts of the solution procedure will be described.

4.1 Reactive tabu search

The routing problem is solved with a reactive tabu search (RTS) based on the work of Nagy et al. (2013). The rough outline of the procedure is depicted in Figure 2. It starts with the initialization of the search (initial solution, tabu list). In each iteration, a neighbour of the current solution s is generated by applying a selected move m_s. Usually, in tabu search algorithms the best non-tabu move is used or a tabu move if it satisfies an aspiration criterion. However, we work with a candidate list CL here, consisting of n_{CL} ($n_{CL} > 1$) moves. In general, the move m_s to be applied to the current solution s is chosen at random from the candidate list. Tabued moves held in the tabu list are expressed in terms of a customer i and a tour t so that i must not be inserted into t for a number of iterations given by the tabu tenure tt.

Reactive elements are included in the tabu list management changing the tabu tenure based on the search progress. In each iteration, a reinitialization of the tabu search can also be triggered. Otherwise, a local optimization procedure is applied to the current solution s after the application of the move m_s. The components of the RTS are described in detail in the following subsections.

```
 1: procedure REACTIVE TABU SEARCH(in: instance data, parameters, out: best solution $s_{best}$)
 2:     initialize tabu search
 3:     $s := s_{best} := s_{init}$          ▷ initialize current solution $s$ and best found solution $s_{best}$
 4:     while stopping criterion is not met do
 5:         determine move $m_s$ for current solution $s$
 6:         $s := s \oplus m_s$                                        ▷ realize move $m_s$
 7:         if $f(s) < f(s_{best})$ then
 8:             $s_{best} := s$                                    ▷ update best found solution
 9:         end if
10:         update tabu list $TL$
11:         if no reinitialization then
12:             apply local optimization to $s$
13:         end if
14:     end while
15: end procedure
```

Figure 2: Reactive Tabu Search

Initialization of the search

Two different construction heuristics are applied alternatively to generate the initial solution in order to test their impact on the performance of the RTS.
On the one hand, we use the modified Sweep heuristic as in Nagy et al. (2013). This heuristic extends the classical Sweep heuristic of Gillett and Miller (1974) by leaving the 20 % of the customers that are closest to the depot out of the procedure to form single-stop tours. In doing so, poor quality solutions should be avoided and these customers should be left to the RTS algorithm to find the best fitting routes (Wassan et al., 2008; Nagy et al., 2013).
In addition, the savings heuristic of Clarke and Wright (1964) is also applied in order to construct initial solutions. In this case, all customers are included into the construction process. Initially, they form single-stop tours and are successively merged according to the savings criterion until no further merging is possible.
At the very beginning of the solution procedure, the tabu search is initialized with an empty tabu list ($TL := \emptyset$), and a tabu tenure $tt := tt_{init}$, $count_{ors} := 0$ and $ma := 0$. The variables $count_{ors}$ and ma serve for the reactive operations (see below).

Neighbourhood structures and moves

The two inter-route move types used in Nagy et al. (2013) are applied here: *Shift* moves remove one customer from one tour and reinsert the customer into another (which can also be an empty tour). *Swap* moves remove two customers from different tours and reinsert them into the tours of the respective other customer. A move consists of one (*Shift*) or two (*Swap*) customer movements. Each customer movement is characterized by a customer to be moved, a source tour, a target tour and a target position. Hence, the swapped customers are not necessarily inserted into the previous (source) position of the respective other customer but can be inserted into any position. Analogously, any position can be considered for the *Shift* moves. For example, shifting customer i from tour t_1 into tour t_2 at position p_1, and shifting customer i from tour t_1 into tour t_2 at position p_2 are two different moves. This approach is contrary to the original approach of Nagy et al. (2013) who always insert a customer into its best position in the target tour. Furthermore, in Nagy et al. (2013), the whole neighbourhood is evaluated, i.e. each customer and each target tour is considered for the *Shift* moves and each pair of customers in different tours for the *Swap* moves. In contrast, we aim to omit potentially unpromising moves in order to save computing time while maintaining the solution quality. As relevant criterion we use the distance $\Delta(t_1, t_2)$ between two tours t_1 and t_2 which is calculated using the minimum and maximum x- and y-values of the coordinates of the customers included in the respective tours:

$$\begin{aligned}\Delta(t_1,t_2) = &\left(\max[\min_{i\in R'_{t_1}} x_i, \min_{i\in R'_{t_2}} x_i] - \min[\max_{i\in R'_{t_1}} x_i, \max_{i\in R'_{t_2}} x_i]\right)\\ &+ \left(\max[\min_{i\in R'_{t_1}} y_i, \min_{i\in R'_{t_2}} y_i] - \min[\max_{i\in R'_{t_1}} y_i, \max_{i\in R'_{t_2}} y_i]\right).\end{aligned}$$

Here, R'_{t_i} stands for the set of customers served in tour t_i $(i = 1, 2)$ excluding the depot $(R'_{t_i} = R_{t_i} \setminus \{0\}$ (see Section 2)). Negative distance values indicate two intersecting tours and offer more potential for improvement than pairs of tours that are further away from each other. Only tp_{max} of all tour pairs with the smallest distances $\Delta(t_1, t_2)$ are considered. tp_{max} is a predefined parameter.

Determining a move for the current solution

In each iteration, a move m_s is determined to construct a new solution from the current one according to $s := s \oplus m_s$. In order to determine m_s, in a first step, a candidate list CL is generated as depicted in Figure 3 (lines 7 to 22).

In the beginning, the set of all moves M for solution s is generated. All of these moves are then examined. In the end, CL contains up to n_{CL} moves m that lead to feasible solutions $s' := s \oplus m$. The feasibility check has the following aspects:

- all tours of a feasible solution s' must not exceed the vehicle weight capacity D and volume capacity V $(V = L \cdot W \cdot H)$,
- feasible packing plans $P_{t,L}$ and $P_{t,B}$ must exist for all tours t of s'. This includes (S5) in the case of SL.

CL contains the best non-tabu moves leading to feasible solutions. In addition, a tabu move m that yields a new best solution $s' := s \oplus m$ with $f(s') < f(s_{best})$ is also accepted for CL, i.e. aspiration by objective is applied.

When the candidate list CL is completely generated, the move m_s is determined as follows:

(1) If CL is empty (including only dummy moves, see Figure 3), the tabued move with the shortest remaining tabu tenure is chosen as m_s, i.e. aspiration by default is used.

(2) If CL is not empty and the solution $s' := s \oplus m_{best}$ (see Figure 3, line 26) is a new best solution, m_s is set to m_{best}; clearly, the best move m_{best} is the move for which $f(s \oplus m)$ gets minimal.

(3) If CL is not empty and there is no new best solution, the move m_s is selected randomly from CL.

Hence, by not necessarily selecting the best neighbour of s, a diversification mechanism is introduced into the search.

```
1: procedure DETERMINE_MOVE(in: current solution $s$, best solution $s_{best}$, instance data, parameters, out: move $m_s$)
2:     initialize set of all moves $M$
3:     $m_d$ := dummy move for initialization of $CL$ with $f(s \oplus m_d) = \infty$
4:     $m_{worst} := m_d$                                        ▷ initialize worst move in $CL$
5:     $m_{TabuShort} := m_d$, $\delta(m_{TabuShort}) := \infty$     ▷ init. move with shortest remaining tabu tenure $\delta$
6:     $CL$ is initialized with $n_{CL}$ dummy moves $m_d$
7:     for all moves $m \in M$ do
8:         $s' := s \oplus m$                                      ▷ generate neighbour $s'$ of $s$
9:         if ($m$ is not tabu and $f(s') < f(s \oplus m_{worst})$
10:            or $f(s') < f(s_{best})$ then                        ▷ aspiration by objective
11:            if $s'$ is feasible then
12:                $CL := CL \cup \{m\}, CL := CL \setminus \{m_{worst}\}$   ▷ current move replaces $m_{worst}$
13:                determine new $m_{worst}$
14:            end if
15:        else
16:            if $m$ is tabu then
17:                if $\delta(m) < \delta(m_{TabuShort})$ then          ▷ aspiration by default
18:                    if $s'$ is feasible then $m_{TabuShort} := m$ end if
19:                end if
20:            end if
21:        end if
22:    end for
23:    if $CL$ contains only dummy moves then
24:        $m_s := m_{TabuShort}$
25:    else
26:        determine best move $m_{best} \in CL$          ▷ $f(s \oplus m_{best}) \leq f(s \oplus m), \forall\ m \in CL$
27:        if $f(s \oplus m_{best}) < f(s_{best})$ then $m_s := m_{best}$
28:        else select $m_s \in CL, m_s \neq m_d$ at random              ▷ without dummy moves
29:        end if
30:    end if
31: end procedure
```

Figure 3: Determination of move m_s

Tabu list management

The tabu list contains all customer movements that are currently tabu, i.e. the information which customers must not be inserted into which tours, as well as the iteration number in which the respective moves are allowed again. The tabu tenure tt determines how long a movement is set tabu. If a *Shift* move removing customer i from tour t was applied in the current iteration, any move that inserts i into t would be tabu for tt iterations. If a *Swap* move was applied, two kinds of movements must be set tabu – one for each customer-tour combination affected: If customer i_1 from tour t_1 was swapped with customer i_2 from tour t_2, any move inserting i_1 into t_1 and i_2 into t_2 is set tabu. That means, that each move inserting i_1 into t_1 *or* i_2 into t_2 is not allowed for the next tt iterations.
Three operations to adapt the search can be carried out within the RTS, namely increasing and decreasing of the tabu tenure tt, and reinitialization of the search. These operations are described in detail in Table 1 (see also Nagy et al., 2013).

Table 1: RTS updating

RTS operation	**Triggering event/ condition**	**Actions**
1. Increasing tabu tenure	(i) currently generated solution $s(it)$ is copy of old solution $s(it\_old)$ **and** (ii) $CopyGap := it - it\_old < GapMax$, i.e. a copy was generated after too few iterations.	- $tt := \min(tt \cdot \phi_{inc}, tt_{max})$ (tt is increased by a factor $\phi_{inc} > 1$, but at most up to tt_{max}) - $ma := 0.1 CopyGap + 0.9 ma$
2. Decreasing tabu tenure	(i) currently generated solution $s(it)$ is not a copy of any older solution **and** (ii) no. of iterations since last tt change is greater than moving average (ma) of $CopyGap$ over all iterations since last (re-)initialization	- $tt := \max(tt \cdot \phi_{dec}, 1)$ (tt is decreased by a factor $\phi_{dec} < 1$, but at most down to 1)
3. Reinitialization of the search	- for each generated solution $s(it)$, $\eta(s)$ is defined as the number of iterations it_old for which $s(it) = s(it\_old)$, i.e. $s(it)$ is a copy of a former solution - if $\eta(s)$ exceeds the limit η_{max}, the counter for often-repeated solutions $count_{ors}$ is increased by one - a reinitialization of the search is triggered if $count_{ors}$ exceeds the threshold $count_{ors,max}$	- $TL := \emptyset$ - $tt, count_{ors}$, and ma are set back to their initial values (see above) - an old solution is selected at random to be the new initial solution

The parameter $GapMax$ is handled in a different fashion compared to Nagy et al. (2013) who assume it to be constant. The results could be improved, though, by taking the size of the instance into account. If an instance contains many customers, there are more possibilities for the algorithm to move around a local optimum and to avoid tabued moves. Therefore, we determine this value as $GapMax = \lambda_{gap} \cdot \sqrt{n}$. In addition, a maximum tabu tenure $tt_{max} = \lambda_{tt} \cdot n$ is considered. λ_{gap} and λ_{tt} are predefined parameters.

Local optimization

At the end of an iteration, two post-optimization procedures are utilized. They are applied only to the tours affected in the current iteration. The first routine is a sequence of intra-route shifts. That is, a customer is moved to another position within the tour if this shift leads to a TTD reduction. This is done iteratively for all customers in the tour until no further improvement is achieved.
In the second routine, the visiting sequence of the customers in a tour is reversed if this reduces the maximum load of a route. In Nagy et al. (2013), the utilization of the capacity is taken into account for this purpose since they consider the one-dimensional case. Here, the maximum loading length needed for linehaul and backhaul items that are simultaneously transported (cf. Figure 1) is reduced if possible. By this means, an additional customer might be visited in the given route.

Tour number reduction

The heuristics for the construction of the initial solution can lead to solutions with more than v_{max} tours. In order to guide the search towards solutions that do not violate the maximum tour number constraint, objective function values of solutions with more than v_{max} tours are penalized. A penalty term $(p \cdot c_{max} \cdot \max(0, v - v_{max}))$ is added to the TTD. p is a fixed parameter, c_{max} is the maximum distance between any two customers ($c_{max} = \max_{(i,j)\in E} c_{ij}$), v is the number of vehicles used in the respective solution. The factors c_{max} and p serve to ensure a sufficiently large penalty.
Furthermore, in connection with the *Shift* move, a customer can only be moved into an empty tour if less than v_{max} tours are used in the current solution.

Stopping criteria

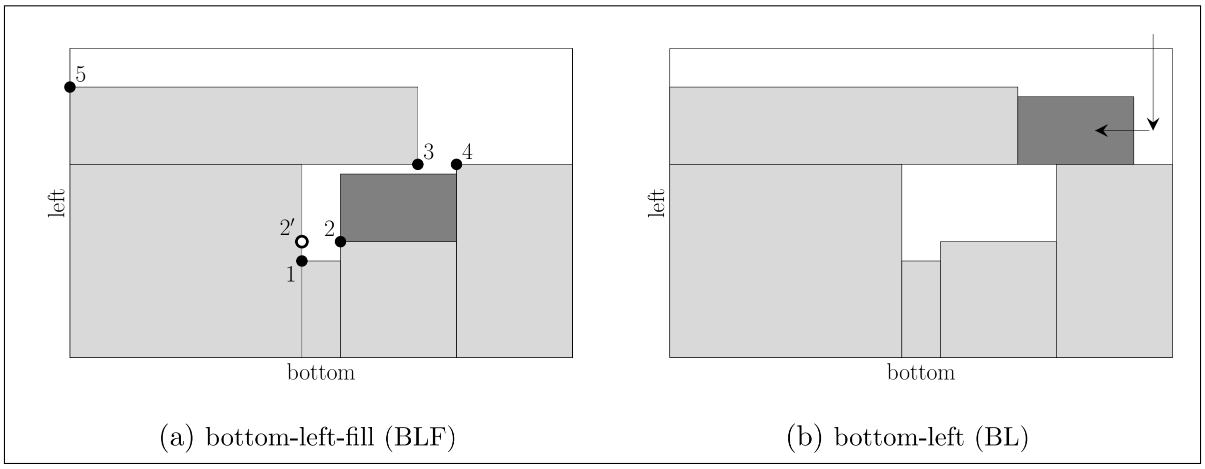
The algorithm has to run for at least $iter_{min}$ iterations. However, the search may continue beyond $iter_{min}$ iterations if the last improvement was less than $iter_{no\_impr}$ ago. Then, it stops after $iter_{no\_impr}$ iterations without improvement.

4.2 Packing heuristics

Three different variants of the deepest-bottom-left-fill (DBLF) heuristic have been integrated into the RTS. It was originally developed for 2D packing (Baker et al., 1980; Hopper, 2000) and extended to 3D packing (Karabulut and Inceoglu, 2005). The DBLF variants are described here for the 3D container loading problem (3D-CLP). In the 3D-CLP a subset of a set of 3D (cuboid) items has to be packed into one container of fixed dimensions such that the filling rate is maximized. Below, we adapt these heuristics to ensure the feasibility of a route w.r.t. the packing subproblem within the 3L-VRPMB.

DBLF heuristics for the 3D container loading problem

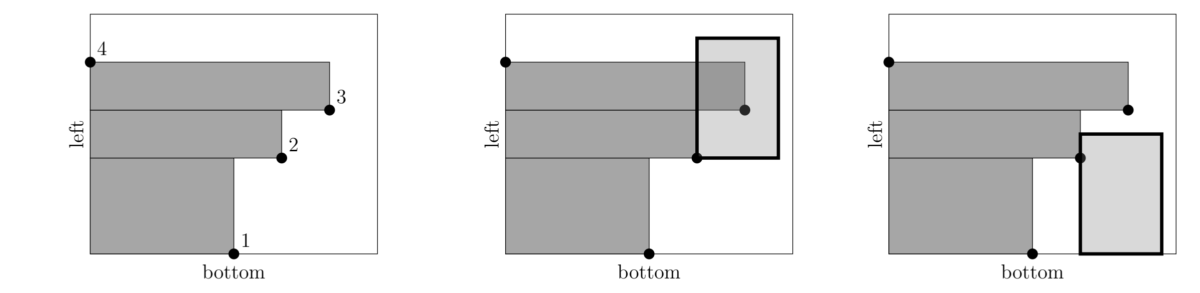
The first variant of the DBLF heuristic that we consider, is based on the DBLF implementation presented by Karabulut and Inceoglu (2005) (and Hopper (2000) for the 2D case). In the approach of Karabulut and Inceoglu (2005) items are packed according to a given sequence. If an item cannot be placed feasibly in a container, it is skipped. Moreover, the spatial orientation is provisionally assumed to be fixed. The priorities for the placement are to position the items as far as possible to the back, (then) to the bottom and (then) to the left of the loading space. In implementations of the DBL heuristic, the final placement is often found using a sliding technique. On the contrary, the *Fill* method allows to fill gaps by keeping track of all possible placement positions and places each item in the deepest, bottom-most, left-most available position. The different approaches are depicted in Figure 4. (For the sake of simplicity, the figures in this section show 2D problems). In the case of the *Fill* approach (Figure 4a), the potential positions are defined by the top-left-back, bottom-right-back, and bottom-left-front corners of already placed items. The positions are sorted based on the deepest-bottom-left priority which is indicated by the numbers 1-5 in the example figure. It is successively tested whether the placement in the respective positions would be feasible. The tests terminate as soon as a feasible position was identified or all position have been tested. The position that is nearest to the bottom and (then) nearest to the left and where the placement would be feasible is position 2. Hence, the gap could be filled which is not possible by sliding the item from the top-right corner (Figure 4b). After a successful placement test for a given position, an item is (if possible) further moved starting from said position as far as possible towards the back, the bottom and the left. In the example in Figure 4a, the final position of the item would be position 2′.



(a) bottom-left-fill (BLF) (b) bottom-left (BL)

Figure 4: Comparison between BLF- and BL-approach

The second implementation variant extends the the placement test of the heuristic presented above and is subsequently called DBLF$^+$. A position where an item cannot be placed feasibly is not considered any further in the DBLF approach. On the contrary, sliding an item into *one* direction is considered during the placement in the variant DBLF$^+$ (regardless of whether a feasible placement is already possible without the sliding). That way, a feasible placement could result from an otherwise infeasible position. The DBLF priority is applied to the sliding, too. That is, first, sliding towards the back, then towards the bottom and then towards the left is tested. If sliding is possible towards one direction, the other ones are not regarded any more. As before, further movements towards the back, bottom and left can be conducted *after* the placement test was successful.
An example is illustrated in Figure 5. Testing the placement in position 2 according to the DBLF approach would lead to an infeasible placement (Figure 5b). The item would overlap with other items. Thus, the next position would be tested now. However, considering sliding for position 2 would lead to a placement further to the bottom which is feasible (Figure 5c).
The third variant is a combination of the original DBLF implementation and DBLF$^+$ in which the sliding of DBLF$^+$ is only used once the original DBLF procedure cannot find a feasible placement for an item. The variant is subsequently called DBLF-Comb and is outlined in Figure 6. The functions *placement_DBLF* and *placement_DBLF$^+$* represent the respective procedure in which the feasibility of a placement is tested.
Furthermore, variants of the touching area (TA) heuristic (e.g. Lodi et al., 1999) were also implemented and tested but led to worse results than the DBLF variants. Therefore,



(a) Placed items with potential positions (b) Without consideration of sliding (c) With consideration of sliding

Figure 5: Illustration of DBLF$^+$

they will not be described in further detail here.

```
1: procedure DBLF-COMB(in: instance data, sequence of items $I$, out: packing plan)
2:     initialize sorted set of positions $P := \{(0,0,0)\}$
3:     for $i := 1$ to $|I|$ do
4:         current item $item := I(i)$
5:         $placed := false$    ▷ binary variable stating whether an item could be placed feasibly or not
6:         for $p := 1$ to $|P|$ do
7:             if placement_DBLF($item, p$) is feasible then $placed := true,\ p' := p$, break end if
8:         end for
9:         if $placed = false$ then
10:            for $p := 1$ to $|P|$ do
11:                if placement_DBLF$^+$($item, p$) is feasible then $placed := true,\ p' := p$, break end if
12:            end for
13:        end if
14:        if $placed = true$ then
15:            place $item$ in $P(p')$ (with sliding if necessary)
16:            move $item$ as far as possible towards the back, bottom, left
17:            update $P$
18:        end if
19:    end for
20: end procedure
```

Figure 6: DBLF-Comb

Adaption of the DBLF heuristics to the 3L-VRPMB

In order to apply the three packing heuristics to the packing subproblem described in Section 2, the following modifications have been made:

- In the 3L-VRPMB, each item has two possible spatial orientations. Therefore, the following placement rule is applied: if placing an item in a given position with the first orientation fails, the same position is tested again with the second orientation.
- Only those placements are accepted where the geometrical constraints ((P1)-(P3)) as well as the packing constraints *vertical stability*, *fragility* and *LIFO* are satisfied.

- In order to facilitate finding a feasible packing plan for a given route, the following sequence of items is chosen: The first items to be placed are the ones of the last customers to be visited (in the case of linehaul customers, vice versa for backhaul customers). The items of one customer are sorted by the item fragility (non-fragile items first), breaking ties by non-increasing volume, breaking ties by non-increasing length and breaking ties by non-increasing width.
- The packing problem to be solved is the orthogonal packing problem (OPP), i.e. the objective is not to maximize the filling rate but to load *all* items belonging to a route feasibly into the loading space. Thus, the procedure is aborted as soon as one item cannot be placed feasibly in any available position.
- For the SL approach, the heuristics were modified. The LIFO restriction as described above must be considered along the width axis of the loading space in order to avoid rearrangements when (un-)loading items. Moreover, it should also be considered along the length axis in order to guarantee that the unloading of linehaul items gradually decreases L_{LH} and, thus, creates space for the backhaul items (see Figure 1). However, simply extending the restriction in this manner resulted in packing patterns as in Figure 7a. Let the visiting sequence here be $\{4, 3, 2, 1\}$. Items of neither customer 3 nor 4 can be placed behind item I_{22} (from the view of the unloading side). In order to avoid such gaps, the placement priorities have been adjusted for the SL: With first priority, an item is to be placed as close as possible towards the origin of the loading space, where the distance is defined as the sum of length and width coordinate of a given placement position. Subsequently, the DBLF rule is applied again, i.e. ties are broken by non-increasing length coordinate, then by height coordinate, and then by width coordinate. This rule results in patterns like in Figure 7b where items tend to be stacked first and then to be arranged towards the side from which they are loaded (cabin or rear side).

4.3 Integration of routing and packing

The packing procedure is executed in order to check whether the items transported on a tour can be packed feasibly according to the above-formulated packing constraints. In the course of a packing check, (generally) two packing plans – one for the linehaul items

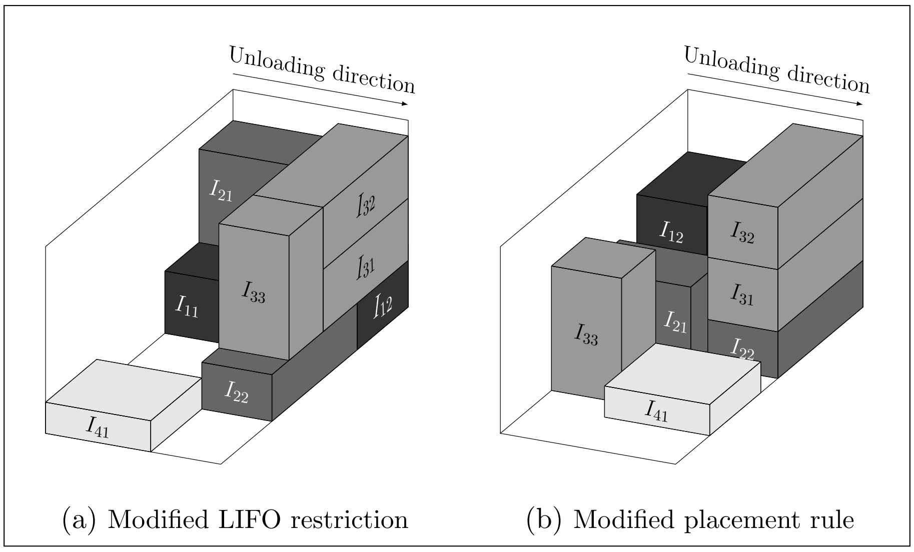


Figure 7: Example packing patterns for side loading

and one for the backhaul items of a tour – are determined. Moreover, in the case of the SL approach, the procedure tests whether linehaul and backhaul items would overlap at any stop of the route according to the generated packing plans (see restriction (S5), Section 2).

A packing check is made when a route is tested for feasibility (see Figure 3). Feasibility tests are conducted whenever a new solution is generated, i.e. during the generation of the initial solution, during the generation of neighbour solutions and in connection with the local optimization. Firstly, it is checked whether the items transported simultaneously exceed the weight and volume capacity of the vehicle at any stop of the tour. If this is not the case, the packing procedure is called. Note, that not every neighbour of the current solutions is tested for feasibility during the TS. Only if the corresponding move can potentially be added to the candidate list, the routes affected by the move are tested. That way, the efficiency is increased significantly (cf. Figure 3, lines 11 and 18).

In the course of the packing procedure, the maximum loading length of a packing plan is determined. This is needed (i) for the tests regarding the SL approach, and (ii) for the second routine of the local optimization (reverse operator).

The application of the packing procedure is usually computationally very expensive. Therefore, a cache is used which stores routes that have already been tested for packing feasibility.

5 Computational experiments

5.1 Benchmark instances

One-dimensional instances

In order to make sure the implemented routing approach is working properly, it was applied to one-dimensional (1D) benchmark instances and the results were compared to the ones obtained by Nagy et al. (2013) and to the best known solutions from the literature. The first instance set (GJB89) was proposed by Goetschalckx and Jacobs-Blecha (1989) for the VRPCB. It consists of 62 instances with 25 to 150 customer locations and 50 %, 66 %, and 80 % linehaul customers. The second set (TV97) was proposed by Toth and Vigo (1997), also for the VRPCB. The 33 instances are based on well-known CVRP instances. The number of customers varies between 21 and 100 and the shares of linehaul customers are the same as in the GJB89 set. Finally, the third set (SN99) was generated by Salhi and Nagy (1999) for the VRPMB. This set consists of instances with 50 to 199 customer locations with 10 %, 25 %, and 50 % backhaul customers. Note that we consider only those 21 instances that do not include drop times and tour length limits.

Three-dimensional instances

Since this paper is – to the best of our knowledge – the first to deal with 3L-VRPMB, we cannot utilize benchmark instances from the literature for the tests with 3D loading constraints. Hence, new instances were generated covering a wide range of different aspects.

The number of customers are set to 20, 60, and 100, and the share of linehaul customers to 50 % and 80 %. The locations of the customers were determined randomly. Moreover, we keep the total number of items fixed to 200 items so that the number of items per customer is varied by the number of customers in an instance (5-15 for $n = 20$, 2-5 for $n = 60$, 1-3 for $n = 100$).

We consider different levels of heterogeneity with respect to the items, that is, instances were generated with 3, 10 or 100 different item types. The edge dimensions and weights of each item type were generated randomly. There are instances with large items and instances with small items. The length and width of large items was uniformly randomly generated in the intervals $[0.2L, 0.6L]$ and $[0.2W, 0.6W]$, respectively (cf. Gendreau et al.,

2006). The height of an item was determined in the interval $[0.2H, 0.5H]$. The factor 0.5 was chosen for the upper limit of the interval to ensure that the items can be placed inside the partitioned loading space (cf. Section 2). For instances with 20 customers, only small items are considered, since all large items of one customer would nearly completely fill the loading space due to the comparatively large number of items per customer. The length (respectively width and height) of small items is generated in the interval $[0.1L, 0.3L]$ (respectively $[0.1W, 0.3W]$ and $[0.1H, 0.3H]$).
The weight of an item type was randomly generated between $1[\frac{weight\ units}{vol.\ units}] \cdot$ item vol.$[vol.\ units]$ and $10[\frac{weight\ units}{vol.\ units}] \cdot$ item vol.$[vol.\ units]$. 20 %, but at least one, of the item types is fragile. The loading space dimensions are $L = 60$, $W = 25$, $H = 30$ and the weight capacity $D = 200$. All in all, 300 different instances were generated for the 3L-VRPMB. The 3L-VRPMB instances as well as our best solutions can be found online at `http://www.mansci.ovgu.de/Forschung/Materialien.html`.

5.2 Parameter setting

The parameter settings were mostly adapted from Nagy et al. (2013). The values for the newly introduced parameters were determined in pre-tests with a set of test instances taken from the three sets for the VRPMB. Their settings are given in Table 2.

Table 2: Parameter settings

Parameter	Value	Parameter	Value	Parameter	Value
$count_{ors,max}$	6	tt_{init}	1	λ_{tt}	4
$iter_{min}$	1500	tp_{max}	30 %	λ_{gap}	50
$iter_{no\_impr}$	200	p	4	ϕ_{inc}	1.1
n_{CL}	3	η_{max}	3	ϕ_{dec}	0.9

5.3 Computational results

The hybrid algorithm was implemented in C++ and the experiments were conducted on a Haswell system with up to 3.2 GHz and 16 GB RAM per core.

One-dimensional instances

Due to some random components in the algorithm, five runs of the RTS were conducted for each instance and with each initial solution heuristic. The RTS in combination with the Savings procedure for the construction of the initial solution will also be referred to as *RTS_Sav* in the following. The combination with the Sweep heuristic shall be called *RTS_Sweep*. The results for the 1D instances are summarized in Table 3. They are subdivided by the applied initial solution heuristic, Sweep or Savings, and by the instance sets.

The column *feas runs* contains the share of feasible solutions among all runs, i.e. where the maximum available tour number v_{max} was not exceeded. The column *feas inst* gives the share of instances that could be solved feasibly at least once.[1] In the following, the obtained solutions are compared to the best solutions of Nagy et al. (2013) (*NWS13*) and the best known solutions (BKS) from the literature (*BKS*). The columns *dev_avg* contain the average percentage deviations of the average obtained TTDs from the solutions of Nagy et al. (2013) and the BKS, respectively. The columns *dev_best* contain the average deviations of the *best* found TTD per instance from the solutions of Nagy et al. (2013) and the BKS, respectively. In the last column the average computing time in seconds is given.

The results are very close to those obtained by the original approach differing on average 0.69 % (*RTS_Sweep*) and -0.12 % (*RTS_Sav*), respectively, w.r.t. the TTD. Note, that Nagy et al. (2013) did not consider the tour number restriction. Therefore, the solutions of about 4.4 % of the instances exceed the tour number limit. Despite the above-mentioned efforts to guide the search towards solutions with feasible tour numbers, this could only be realized for 93.3 % of the runs by our implementation and three of the instances could not be solved feasibly at all.

With an average deviation from the BKS of 1.47 %, *RTS_Sav* produces better results than *RTS_Sweep* which led to an average deviation from the BKS of 2.34 %. As we were able to improve the results obtained by Nagy et al. (2013) by applying the Savings heuristic, it seems the choice of the initial solution construction heuristic has a significant impact on the performance of the RTS. Furthermore, slightly less computing time is needed for *RTS_Sav* (on average about 4 minutes) than for *RTS_Sweep* (4.3 min).

The computing times depend heavily on the instance size, i.e. the number of customers. The instances examined here contain 21 to 199 customers and the computing times range from less than one second to about 40 minutes. Thus, they exceed the ones of Nagy et al.

[1]Note, that for the instance set SN99 no tour number restriction is given. Hence, all solutions are feasible.

(2013) who state that less than a minute was needed for the majority of the instances. Nevertheless, the computing times for both variants of the algorithm are tolerable.

Table 3: RTS results for one-dimensional instances

Init/ Set	#	**feas** *runs*	*inst*	**dev_avg [%]** *NWS13*	*BKS*	**dev_best [%]** *NWS13*	*BKS*	**t[s]**
Sweep	**116**	**91.6**	**95.7**	**0.69**	**2.34**	**−0.15**	**1.48**	**257.9**
GJB89	62	91.0	96.8	1.20	2.12	0.15	1.06	257.3
TV97	33	87.3	90.9	0.01	0.94	−0.59	0.33	89.6
SN99	21	100.0	100.0	0.25	5.17	−0.34	4.54	523.8
Savings	**116**	**95.0**	**96.6**	**−0.12**	**1.47**	**−0.68**	**0.91**	**239.1**
GJB89	62	96.8	96.8	0.24	1.14	−0.26	0.63	223.1
TV97	33	88.5	93.9	0.23	1.16	−0.47	0.45	108.4
SN99	21	100.0	100.0	−1.75	2.96	−2.24	2.45	491.6

Moreover, taking into account only the best solutions achieved with *any* of the initial solution procedures, the average deviation from the BKS amounts to only 0.37 % and the average deviation from the best results obtained by Nagy et al. (2013) amounts to −1.22 %. Those results are shortly summarized in Table 4. Regarding the computing times, that would mean that the time consume would still be below ten minutes (on average) if both variants are applied to an instance.

Table 4: RTS results for the best solutions obtained (1D)

Set	**dev NWS13[%]**	**dev BKS[%]**
GJB89	−0.83	0.06
TV97	−0.92	−0.01
SN99	−2.81	1.86
Total	−1.22	0.37

Three-dimensional instances

As the final hybrid algorithm must be equipped with a packing heuristic, the available packing heuristics where initially tested and compared. For this purpose, a test set of 30 out of the 300 instances was formed. The parameter settings determined with the one-dimensional instances where also applied to the 3D case.

Determination of the best packing heuristic

Table 5 contains the results of comparing the different packing heuristics. Since no benchmark results are available for the newly generated instances, the obtained results are compared to the best found solutions over all heuristics, loading approaches and all five runs. The table contains the average (*avg dev*) and maximum (*max dev*) percentage deviation of the obtained TTDs from the best found ones. The results are presented for both loading approaches, loading space partition (*LSP*) and side loading (*SL*).

The best results were obtained with the SL approach and the DBLF heuristic. However, in combination with the LSP the combination of the DBLF with DBLF^{+} produced better results than the *plain* DBLF heuristic. Hence, the respective results were examined in more detail with the conclusion that the DBLF seems to work better for instances with small items whereas the combination dominates in the case of large items. Therefore, in the final tests, the packing heuristic is chosen depending on the item sizes: DBLF for small items and DBLF-Comb for large items.

Table 5: Comparison of packing heuristics

	avg dev[%]		**max dev[%]**	
Packing heuristic	*LSP*	*SL*	*LSP*	*SL*
DBLF	18.7	1.3	44.8	6.7
DBLF^{+}	18.1	5.0	42.3	10.8
DBLF-Comb	17.0	2.6	42.3	8.5

Results for the hybrid algorithm

The final experiments were conducted with all 300 instances each of which were solved five times by the RTS for both loading approaches and with both initial solution construction heuristics. Hence, 6,000 runs were performed in total.

Two aspects should be analysed in the following: (i) the impact of the initial solution construction heuristic and (ii) the impact of the loading approach. The results regarding the first aspect are summarized in Table 6. The results obtained with *RTS_Sav* and the respective loading approaches serve as a benchmark here. The columns *avg dev* contains the average percentage deviations of the TTDs obtained with *RTS_Sweep* from the average benchmark TTDs aggregated over both loading approaches.[2] The results support the findings from the 1D tests (Table 3) indicating that *RTS_Sav* slightly outperforms *RTS_Sweep*. The TTDs obtained with *RTS_Sweep* deviate on average 0.55 % from the TTDs obtained with *RTS_Sav*.

Table 7 provides a comparison of the two different loading approaches. Here, the results

[2]The *average benchmark TTDs* refer to the average of the TTDs obtained with the above-mentioned benchmark procedure over all five runs.

Table 6: RTS results for 3D instances: Initial solution construction heuristics

Instances		**avg dev[%]**
		RTS_Sweep
n	*Items*	*from RTS_Sav*
20	small	0.05
60	large	0.13
	small	0.36
100	large	1.20
	small	0.98
Total		0.55

obtained with LSP and the respective initial solution heuristic were used as benchmark. In the table, the average deviations (*avg dev*) of the SL solutions (w.r.t. the TTDs) from the LSP solutions (aggregated over both initial solution heuristics) and the average number of used vehicles (v) are reported. Furthermore, the average computing times (in seconds) are reported in the columns t. Not surprisingly, much better results can be obtained with the SL approach. The respective TTDs are on average almost 14 % lower than the TTDs obtained with the LSP and less vehicles are needed. Furthermore, for 262 out of the 300 instances (among them are *all* instances with large items) the best solutions were found with the SL approach. These results could be expected as the SL approach allows making use of the whole loading space whereas half of it is left empty at least at the very beginning and the very end of a tour if the loading space is separated. However, differences can be observed among the different instance classes. The SL approach seems to be particularly beneficial if the items are large (as smaller items can be placed more easily also within a smaller loading space), and if the share of linehaul and backhaul items is unequal. Moreover, the differences decrease with an increasing number of customers, i.e. with a decreasing number of items per customer. For example, in the case of $n = 100$, small items and 50 % linehaul customers, the average deviations from the benchmark TTDs are very low and the average tour numbers are only slightly smaller for the SL approach.

Moreover, the RTS with the LSP requires less computing time than the RTS with the SL approach. Two reasons can be identified: On the one hand, checking feasibility for the SL approach takes longer because not only are two packing plans to be generated, but these packing plans also need to be compared for each stop of a tour to avoid any overlapping of linehaul and backhaul items. On the other hand, the SL approach allows

Table 7: RTS results for 3D instances: Loading approaches

Instances			**avg dev[%]** *of SL*	**v**		**t[s]**	
n	*Items*	*LH*	*from LSP*	*LSP*	*SL*	*LSP*	*SL*
20	small	50%	−5.39	2.8	2.1	8.1	22.3
		80%	−11.37	3.6	2.5	6.0	23.6
60	large	50%	−28.99	20.3	11.8	10.7	15.9
		80%	−35.32	30.5	16.1	12.5	12.8
	small	50%	−0.84	2.9	2.6	246.9	286.7
		80%	−4.57	3.5	2.7	150.9	247.6
100	large	50%	−20.33	19.0	13.0	43.7	71.3
		80%	−30.88	29.2	16.6	32.3	53.9
	small	50%	0.13	2.9	2.8	1,778.2	1,719.0
		80%	−2.36	3.6	2.9	1,088.1	1,606.4
Total			−13.93	11.8	7.3	337.7	406.0

to form longer tours (see below) which also require a higher packing effort. Independently of the loading approach, it can be observed that the computing times increase with the size of the underlying VRP, i.e. with the number of customers, and that the computing times depend on the item sizes. Instances with large items can be solved much faster than instances with small items. In the case of small items, much more customers can be merged into one tour (resulting in more items per tour) which requires higher efforts for the packing checks.

6 Conclusions

Transporting linehaul and backhaul goods in the same tours can be a useful mean to reduce empty running trucks, travelled distances, fuel consumption and, in consequence, to reduce costs. Although the integration of backhauls into the VRP was studied frequently in the research, the transported goods and available capacities were mostly considered to be one-dimensional. Applying solutions obtained by solving such problems could turn out to be infeasible when the transported goods are bulky. Therefore, we present a 3L-VRPMB which includes not only the backhauls into the VRP but also three-dimensional loading constraints so that, for example, load stability or loading sequences can be considered. In order to solve it, a hybrid algorithm consisting of a reactive tabu search (which was originally developed for the one-dimensional VRPMB) combined with a packing construction heuristic was implemented.

The procedure was tested for both one-dimensional and three-dimensional test instances. The one-dimensional tests indicate that the implemented RTS is comparable to the original approach. Moreover, a second heuristic for constructing initial solutions was applied which led to further improvements of the results.

New instances were generated for the 3L-VRPMB since there are no benchmark instances available from the literature. Different packing construction heuristics were implemented and compared. The final hybrid algorithm combined the RTS with variants of the deepest-bottom-left-fill approach. Since linehaul and backhaul customers can be visited in any sequence in the VRPMB, linehaul and backhaul items are (partly) transported simultaneously. In the hybrid approach, this is realized by two different loading approaches which ensure that any reloading during a tour is avoided. They include rear-loaded vehicles with horizontally separated loading spaces into linehaul and backhaul sections, and side-loaded vehicles. The best results – both in terms of total travel distance and number of tours – were obtained with the side loading approach. However, the loading space partition is implemented easier and requires, on average, less computing time.

Integrating backhauls into 3L-VRPs and, in particular, dealing with the simultaneous transport of delivery and pickup products are very interesting topics. Further VRP variants which include such a problem are, for example, the VRP with simultaneous delivery and pickup or the pickup and delivery problem. Whereas reloading had to be avoided in the problem variants presented here, allowing reloading during a tour could also be a promising approach. This way, more tours might be feasible and the travelling costs could be further reduced.

Appendix

Followingly, the detailed results of the one-dimensional tests are presented. Tables 8 to 10 state the instance characteristics in the first columns. The column *BKS* contains the best known solution from the literature with the respective reference in the column *Ref.* The next columns contain the average (*avg*) and the best found solutions (*best*) of our RTS and the number of vehicles corresponding to the best solution. The last column states whether the best solution was obtained in connection with the Sweep (*Sw*) heuristic, the Saving (*Sav*) heuristic, or both.

The references are the following: H92 - Halse (1992), W02 - Wade and Salhi (2002), WS04 - Wade and Salhi (2004), RP06 - Ropke and Pisinger (2006), WNA08 - Wassan et al. (2008), TCB09 - Tütüncü et al. (2009), JK12 - Jun and Kim (2012), NWS13 - Nagy et al. (2013), GBG15 - García-Nájera et al. (2015).

Table 8: Results RTS for GJB89 instance set

Inst.	**n**	**LH[%]**	$\mathbf{v_{max}}$	**BKS**	**v**	**Ref**	**avg**	**best**	**v**	**init**
A1	25	80	8	223,088	8	WS04	223,325.32	223,088	8	both
A2	25	80	5	169,450	5	WNA08	169,499.78	169,500	5	both
A3	25	80	4	141,984	3	WNA08	142,033.89	142,034	3	both
A4	25	80	3	141,984	3	WNA08	142,033.89	142,034	3	both
B1	30	66	7	232,371	7	WNA08	233,039.31	232,436	7	both
B2	30	66	5	179,194	4	WNA08	179,944.00	179,194	4	Sav
B3	30	66	3	145,583	3	WNA08	145,701.96	145,702	3	both
C1	40	50	7	237,100	7	WNA08	238,178.31	237,110	7	both
C2	40	50	5	196,683	5	WNA08	198,543.21	196,683	5	Sw
C3	40	50	5	164,794	3	WNA08	166,673.33	164,891	3	both
C4	40	50	4	164,794	3	WNA08	167,307.77	164,891	3	Sw
D1	38	80	12	307,109	11	WNA08	307,109.60	307,110	11	both
D2	38	80	11	307,109	11	WNA08	307,109.60	307,110	11	both
D3	38	80	7	220,700	7	WNA08	221,523.54	220,751	7	both
D4	38	80	5	182,496	5	NWS13	185,686.79	182,928	5	both
E1	45	66	7	220,742	7	WNA08	222,093.97	220,742	7	both
E2	45	66	4	190,048	4	H92	191,253.72	190,049	4	Sav
E3	45	66	4	182,804	4	WS04	183,125.52	**181,941**	4	Sav
F1	60	50	6	243,599	6	NWS13	248,266.23	244,353	6	Sav
F2	60	50	7	243,599	6	NWS13	248,655.93	244,353	6	Sav
F3	60	50	5	212,296	4	NWS13	213,704.38	212,296	4	Sw
F4	60	50	4	200,964	4	WS04	203,940.59	**198,709**	4	Sav
G1	57	80	10	297,707	10	WNA08	302,177.04	**297,656**	9	Sav
G2	57	80	6	234,653	6	NWS13	234,620.30	**234,101**	6	Sw
G3	57	80	5	213,757	5	H92	215,246.11	**212,748**	5	both
G4	57	80	6	213,757	5	H92	215,048.95	**212,748**	5	both
G5	57	80	5	202,610	4	H92	204,412.95	**200,521**	4	Sav
G6	57	80	4	188,823	3	NWS13	193,181.71	**188,696**	3	Sw
H1	68	66	6	235,269	6	H92	238,854.46	236,427	6	Sav
H2	68	66	5	214,908	5	WNA08	218,519.64	**213,732**	5	Sav
H3	68	66	4	202,971	4	H92	205,365.49	204,794	4	both
H4	68	66	5	202,971	4	H92	205,287.93	204,794	4	both
H5	68	66	4	201,896	4	H92	202,399.25	**196,446**	4	Sav
H6	68	66	5	201,896	4	H92	202,206.22	**196,446**	4	Sav
I1	90	50	10	320,703	9	WNA08	322,052.95	**320,217**	10	Sav
I2	90	50	7	272,621	7	NWS13	279,804.88	276,519	7	Sav
I3	90	50	5	238,245	5	NWS13	245,919.54	**237,662**	5	Sw
I4	90	50	6	238,245	5	NWS13	246,280.38	**237,662**	5	Sw
I5	90	50	7	238,245	5	NWS13	246,303.65	**237,662**	5	Sw
J1	94	80	10	330,235	10	NWS13	327,558.23	**324,265**	10	Sav
J2	94	80	8	292,698	8	WS04	297,700.88	294,004	8	Sav

Table 8: Results RTS for GJB89 instance set (*continued*)

Inst.	**n**	**LH[%]**	$\mathbf{v_{max}}$	**BKS**	**v**	**Ref**	**avg**	**best**	**v**	**init**
J3	94	80	6	249,931	6	WNA08	261,643.05	255,195	6	Sav
J4	94	80	7	257,895	6	H92	282,182.11	275,311	7	Sav
K1	113	66	10	352,253	10	WNA08	357,108.88	352,729	10	Sw
K2	113	66	8	317,004	8	WNA08	326,955.11	317,274	8	Sav
K3	113	66	9	317,004	8	WNA08	326,983.96	317,562	8	Sav
K4	113	66	7	294,848	7	NWS13	297,758.79	**293,621**	7	Sw
L1	150	50	10	394,414	10	WNA08	406,513.44	395,803	10	Sav
L2	150	50	8	360,018	8	WS04	372,626.10	365,189	9	Sw
L3	150	50	9	360,018	8	WS04	372,626.10	365,189	9	Sw
L4	150	50	7	337,620	7	WS04	345,706.84	**335,186**	7	Sav
L5	150	50	8	337,620	7	WS04	346,031.85	**335,186**	7	Sav
M1	125	80	11	360,897	10	WNA08	372,174.16	366,426	10	Sav
M2	125	80	10	360,897	10	WNA08	372,405.47	364,870	10	Sav
M3	125	80	9	335,486	9	WS04	340,686.24	336,753	9	Sw
M4	125	80	7	300,225	7	NWS13	307,637.90	305,428	7	Sav
N1	150	66	11	370,690	10	WS04	370,682.35	**365,724**	10	Sav
N2	150	66	10	370,690	10	WS04	370,557.85	**365,583**	10	Sav
N3	150	66	9	349,516	9	WS04	362,204.84	352,064	9	Sav
N4	150	66	10	349,516	9	WS04	361,061.15	**348,956**	9	Sav
N5	150	66	7	319,811	7	H92	328,827.84	**315,475**	7	Sav
N6	150	66	8	319,811	7	H92	328,380.55	**315,988**	7	Sav

Table 9: Results RTS for TV97 instance set

Inst.	**n**	**LH[%]**	$\mathbf{v_{max}}$	**BKS**	**v**	**Ref**	**avg**	**best**	**v**	**init**
eil22_50	21	50	3	324	3	NWS13	326.02	326	3	both
eil22_66	21	66	3	341	3	NWS13	341.91	342	3	both
eil22_80	21	80	3	341	3	NWS13	342.23	342	3	both
eil23_50	22	50	2	526	2	W02	527.73	527	2	both
eil23_66	22	66	2	526	2	NWS13	529.53	527	2	both
eil23_80	22	80	2	514	2	W02	513.91	514	2	both
eil30_50	29	50	2	417	2	NWS13	421.52	419	2	both
eil30_66	29	66	3	475	3	NWS13	501.35	484	3	Sav
eil30_80	29	80	3	475	3	W02	482.58	478	3	Sw
eil33_50	32	50	3	680	3	NWS13	682.67	681	3	Sw
eil33_66	32	66	3	680	3	NWS13	683.59	681	3	both
eil33_80	32	80	3	686	3	NWS13	712.07	687	3	Sw
eil51_50	50	50	3	466	3	W02	470.12	465	3	Sw
eil51_66	50	66	4	491	4	WNA08	493.56	491	4	Sw
eil51_80	50	80	4	497	4	WNA08	500.33	497	4	Sav
eilA101_50	100	50	4	730	5	WNA08	744.23	739	4	Sw
eilA101_66	100	66	6	754	6	W02	766.88	756	6	Sw
eilA101_80	100	80	6	789	7	W02	807.57	795	6	Sav
eilA76_50	75	50	6	670	6	WNA08	672.69	**666**	6	Sav

Table 9: Results RTS for TV97 instance set (*continued*)

Inst.	**n**	**LH[%]**	$\mathbf{v_{max}}$	**BKS**	**v**	**Ref**	**avg**	**best**	**v**	**init**
eilA76_66	75	66	7	719	7	WNA08	713.48	**704**	7	Sav
eilA76_80	75	80	8	759	8	TCB09	766.26	761	8	Sav
eilB101_50	100	50	7	861	8	WNA08	867.27	**854**	8	Sw
eilB101_66	100	66	9	923	10	WNA08	940.88	931	10	Sw
eilB101_80	100	80	11	969	11	WNA08	990.00	**966**	11	Sav
eilB76_50	75	50	8	768	8	WNA08	767.14	**757**	8	Sav
eilB76_66	75	66	10	826	10	W02	835.27	831	10	Sav
eilB76_80	75	80	12	904	12	WNA08	911.91	**897**	12	Sw
eilC76_50	75	50	5	629	5	WNA08	635.77	**626**	5	Sw
eilC76_66	75	66	6	663	6	W02	663.54	**660**	6	Sw
eilC76_80	75	80	7	697	7	WNA08	700.77	**695**	7	Sav
eilD76_50	75	50	4	608	4	NWS13	611.13	**602**	4	Sw
eilD76_66	75	66	5	627	5	WNA08	637.18	627	5	Sw
eilD76_80	75	80	6	653	6	WNA08	654.59	**646**	6	Sav

Table 10: Results RTS for SN99 instance set

Inst.	**n**	**LH[%]**	**BKS**	**v**	**Ref**	**avg**	**best**	**v**	**init**
CMT01H	50	50	462	3	JK12	471.09	466	3	Sw
CMT01Q	50	75	490	5	RP06	498.64	490	4	Sw
CMT01T	50	90	520	5	RP06	523.67	520	5	Sw
CMT02H	75	50	661	6	JK12	671.84	666	6	Sw
CMT02Q	75	75	732	12	RP06	743.37	733	8	Sw
CMT02T	75	90	783	12	RP06	792.45	785	9	Sw
CMT03H	100	50	701	10	RP06	741.19	723	5	Sw
CMT03Q	100	75	747	10	RP06	770.54	754	6	Sav
CMT03T	100	90	798	10	RP06	814.21	802	7	Sav
CMT04H	150	50	829	14	RP06	876.44	870	7	Sav
CMT04Q	150	75	915	9	JK12	939.15	932	9	Sav
CMT04T	150	90	993	11	GBG15	1,028.29	1,023	11	Sav
CMT05H	199	50	983	20	RP06	1,053.18	1,041	10	Sav
CMT05Q	199	75	1,118	20	RP06	1,171.70	1,157	12	Sav
CMT05T	199	90	1,227	20	RP06	1,295.00	1,280	16	Sw
CMT11H	120	50	818	8	RP06	892.47	847	4	Sav
CMT11Q	120	75	939	8	RP06	1,006.28	944	6	Sav
CMT11T	120	90	999	8	RP06	1,101.16	1,007	7	Sav
CMT12H	100	50	629	12	RP06	651.71	638	5	Sw
CMT12Q	100	75	729	12	RP06	760.62	749	8	Sav
CMT12T	100	90	788	12	RP06	801.70	788	9	Sav

References

Avci, M.; Topaloglu, S. (2015): An adaptive local search algorithm for vehicle routing problem with simultaneous and mixed pickups and deliveries. In: *Computers & Industrial Engineering*, 83, pp. 15–29.

Baker, B. S.; Coffman, Jr., E. G.; Rivest, R. L. (1980): Orthogonal Packings in Two Dimensions. In: *SIAM Journal on Computing*, 9, 4, pp. 846–855.

Bartók, T.; Imreh, C. (2011): Pickup and Delivery Vehicle Routing with Multidimensional Loading Constraints. In: *Acta Cybernetica*, 20, 1, pp. 17–33.

Bortfeldt, A. (2012): A hybrid algorithm for the capacitated vehicle routing problem with three-dimensional loading constraints. In: *Computers & Operations Research*, 39, 9, pp. 2248–2257.

Bortfeldt, A.; Hahn, T.; Männel, D.; Mönch, L. (2015): Hybrid algorithms for the vehicle routing problem with clustered backhauls and 3D loading constraints. In: *European Journal of Operational Research*, 243, 1, pp. 82–96.

Casco, D. O.; Golden, B. L.; Wasil, E. A. (1988): Vehicle routing with backhauls: Models, algorithms, and case studies. In: B. L. Golden, ed., *Vehicle routing*, Studies in management science and systems, pp. 127–147, North-Holland, Amsterdam u.a.

Clarke, G.; Wright, J. W. (1964): Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. In: *Operations Research*, 12, 4, pp. 568–581.

Crispim, J.; Brandão, J. (2005): Metaheuristics Applied to Mixed and Simultaneous Extensions of Vehicle Routing Problems with Backhauls. In: *The Journal of the Operational Research Society*, 56, 11, pp. 1296–1302.

de Angelis, L. (2011): A fall in average vehicle loads: Average loads, distances and empty running in road freight transport-2010. In: *Statistics in Focus*, 63.

Dominguez, O.; Guimarans, D.; Juan, A. A. (2015): A Hybrid Heuristic for the 2L-VRP with Clustered Backhauls. In: *Proceedings of the XVI Conferencia de la Asociación Española para la Inteligencia Artificial (CAEPIA)*.

Escobar-Falcón, L. M.; Álvarez-Martínez, D.; Granada-Echeverri, M.; Escobar, J. W.; Romero-Lázaro, R. A. (2016): A matheuristic algorithm for the three-dimensional loading capacitated vehicle routing problem (3L-CVRP). In: *Revista Facultad de Ingeniería Universidad de Antioquia*, 78, pp. 9–20.

Fuellerer, G.; Doerner, K. F.; Hartl, R. F.; Iori, M. (2010): Metaheuristics for vehicle routing problems with three-dimensional loading constraints. In: *European Journal of Operational Research*, 201, 3, pp. 751–759.

García-Nájera, A.; Bullinaria, J. A.; Gutiérrez-Andrade, M. A. (2015): An evolutionary approach for multi-objective vehicle routing problems with backhauls. In: *Computers & Industrial Engineering*, 81, pp. 90–108.

Gendreau, M.; Iori, M.; Laporte, G.; Martello, S. (2006): A Tabu Search Algorithm for a Routing and Container Loading Problem. In: *Transportation Science*, 40, 3, pp. 342–350.

Gillett, B. E.; Miller, L. R. (1974): A Heuristic Algorithm for the Vehicle-Dispatch Problem. In: *Operations Research*, 22, 2, pp. 340–349.

Goetschalckx, M.; Jacobs-Blecha, C. (1989): The vehicle routing problem with backhauls. In: *European Journal of Operational Research*, 42, 1, pp. 39–51.

Golden, B. L.; Baker, E.; Alfaro, J.; Schaffer, J. (1985): The vehicle routing problem with backhauling: two approaches. In: Hammesfahr; RD, eds., *Proceedings of the twenty-first annual meeting of the S.E. TIMS, Myrtle Beach, SC, USA*.

Halse, K. (1992): *Modeling and solving complex vehicle routing problems. Ph.D. Thesis*, Technical University of Denmark, Lyngby.

Hopper, E. (2000): *Two-dimensional packing utilising evolutionary algorithms and other meta-heuristic methods*. Ph.D. thesis, University of Wales. Cardiff.

Irnich, S.; Toth, P.; Vigo, D. (2014): The Family of Vehicle Routing Problems. In: P. Toth; D. Vigo, eds., *Vehicle routing*, MOS-SIAM series on optimization, pp. 1–33, SIAM, Philadelphia, Pa.

Jun, Y.; Kim, B.-I. (2012): New best solutions to VRPSPD benchmark problems by a perturbation based algorithm. In: *Expert Systems with Applications*, 39, 5, pp. 5641–5648.

Karabulut, K.; Inceoglu, M. M. (2005): A Hybrid Genetic Algorithm for Packing in 3D with Deepest Bottom Left with Fill Method. In: T. Yakhno, ed., *Advances in Information Systems*, vol. 3261 of *Lecture Notes in Computer Science*, pp. 441–450, Springer-Verlag Berlin/Heidelberg, Berlin, Heidelberg.

Lodi, A.; Martello, S.; Vigo, D. (1999): Heuristic and Metaheuristic Approaches for a Class of Two-Dimensional Bin Packing Problems. In: *INFORMS Journal on Computing*, 11, 4, pp. 345–357.

Ma, H.-w.; Zhu, W.; Xu, S. (2011): Research on the Algorithm for 3L-CVRP with Considering the Utilization Rate of Vehicles. In: R. Chen, ed., *Intelligent Computing and Information Science*, vol. 134 of *Communications in Computer and Information Science*, pp. 621–629, Springer Berlin Heidelberg, Berlin, Heidelberg.

Männel, D.; Bortfeldt, A. (2016): A hybrid algorithm for the vehicle routing problem with pickup and delivery and three-dimensional loading constraints. In: *European Journal of Operational Research*, 254, 3, pp. 840–858.

Miao, L.; Ruan, Q.; Woghiren, K.; Ruo, Q. (2012): A hybrid genetic algorithm for the vehicle routing problem with three-dimensional loading constraints. In: *RAIRO - Operations Research*, 46, 1, pp. 63–82.

Moura, A. (2008): A Multi-Objective Genetic Algorithm for the Vehicle Routing with Time Windows and Loading Problem. In: A. Bortfeldt; J. Homberger; H. Kopfer; G. Pankratz; R. Strangmeier, eds., *Intelligent Decision Support*, Gabler Edition Wissenschaft, pp. 187–201, Betriebswirtschaftlicher Verlag Dr. Th. Gabler / GWV Fachverlage GmbH Wiesbaden, Wiesbaden.

Moura, A.; Oliveira, J. F. (2009): An integrated approach to the vehicle routing and container loading problems. In: *OR Spectrum*, 31, 4, pp. 775–800.

Nagy, G.; Wassan, N. A.; Salhi, S. (2013): The vehicle routing problem with restricted mixing of deliveries and pickups. In: *Journal of Scheduling*, 16, 2, pp. 199–213.

Parragh, S. N.; Doerner, K. F.; Hartl, R. F. (2008): A survey on pickup and delivery problems. In: *Journal für Betriebswirtschaft*, 58, 1, pp. 21–51.

Pinto, T.; Alves, C.; De, C.; Moura, A. (2015): An insertion heuristic for the capacitated vehicle routing problem with loading constraints and mixed linehauls and backhauls. In: *FME Transaction*, 43, 4, pp. 311–318.

Pollaris, H.; Braekers, K.; Caris, A.; Janssens, G. K.; Limbourg, S. (2015): Vehicle routing problems with loading constraints: State-of-the-art and future directions. In: *OR Spectrum*, 37, 2, pp. 297–330.

Ropke, S.; Pisinger, D. (2006): A unified heuristic for a large class of Vehicle Routing Problems with Backhauls. In: *European Journal of Operational Research*, 171, 3, pp. 750–775.

Salhi, S.; Nagy, G. (1999): A Cluster Insertion Heuristic for Single and Multiple Depot Vehicle Routing Problems with Backhauling. In: *The Journal of the Operational Research Society*, 50, 10, pp. 1034.

Tao, Y.; Wang, F. (2015): An effective tabu search approach with improved loading algorithms for the 3L-CVRP. In: *Computers & Operations Research*, 55, pp. 127–140.

Tarantilis, C. D.; Zachariadis, E. E.; Kiranoudis, C. T. (2009): A Hybrid Metaheuristic Algorithm for the Integrated Vehicle Routing and Three-Dimensional Container-Loading Problem. In: *IEEE Transactions on Intelligent Transportation Systems*, 10, 2, pp. 255–271.

Toth, P.; Vigo, D. (1997): An Exact Algorithm for the Vehicle Routing Problem with Backhauls. In: *Transportation Science*, 31, 4, pp. 372–385.

Toth, P.; Vigo, D., eds. (2014): *Vehicle routing: Problems, methods, and applications*, vol. 18 of *MOS-SIAM series on optimization*. SIAM, Philadelphia, Pa., 2. ed. edn.

Tütüncü, G. Y.; Carreto, C. A. C.; Baker, B. M. (2009): A visual interactive approach to classical and mixed vehicle routing problems with backhauls. In: *Omega*, 37, 1, pp. 138–154.

Wade, A.; Salhi, S. (2004): An Ant System Algorithm for the Mixed Vehicle Routing Problem with Backhauls. In: *Metaheuristics: Computer Decision-Making*, pp. 699–719, Springer US, Boston, MA.

Wade, A. C.; Salhi, S. (2002): An investigation into a new class of vehicle routing problem with backhauls. In: *Omega*, 30, 6, pp. 479–487.

Wang, L.; Guo, S.; Chen, S.; Zhu, W.; Lim, A. (2010): Two Natural Heuristics for 3D Packing with Practical Loading Constraints. In: B.-T. Zhang; M. A. Orgun, eds., *PRICAI 2010: trends in artificial intelligence*, vol. 6230 of *Lecture notes in computer science Lecture notes in artificial intelligence*, pp. 256–267, Springer, Berlin.

Wassan, N. A.; Nagy, G.; Ahmadi, S. (2008): A heuristic method for the vehicle routing problem with mixed deliveries and pickups. In: *Journal of Scheduling*, 11, 2, pp. 149–161.

Wassan, N. A.; Salhi, S.; Nagy, G.; Wassan, N.; Wade, A. C. (2013): Solving the Mixed Backhauling Vehicle Routing Problem with Ants. In: *International Journal of Energy Optimization and Engineering*, 2, 2, pp. 62–77.

Wei, L.; Zhang, Z.; Lim, A. (2014): An Adaptive Variable Neighborhood Search for a Heterogeneous Fleet Vehicle Routing Problem with Three-Dimensional Loading Constraints. In: *IEEE Computational Intelligence Magazine*, 9, 4, pp. 18–30.

Wisniewski, M. A.; Ritt, M.; Buriol, L. S. (2011): A tabu search algorithm for the capacitated vehicle routing problem with three-dimensional loading constraints. In: *XLIII Simposio Brasilero de Pesquisa Operacional*.

Zachariadis, E. E.; Tarantilis, C. D.; Kiranoudis, C. T. (2016): The Vehicle Routing Problem with Simultaneous Pick-ups and Deliveries and Two-Dimensional Loading Constraints. In: *European Journal of Operational Research*, 251, 2, pp. 369–386.

Zhang, Z.; Wei, L.; Lim, A. (2015): An evolutionary local search for the capacitated vehicle routing problem minimizing fuel consumption under three-dimensional loading constraints. In: *Transportation Research Part B: Methodological*, 82, pp. 20–35.

Zhu, W.; Qin, H.; Lim, A.; Wang, L. (2012): A two-stage tabu search algorithm with enhanced packing heuristics for the 3L-CVRP and M3L-CVRP. In: *Computers & Operations Research*, 39, 9, pp. 2178–2195.

Otto von Guericke University Magdeburg
Faculty of Economics and Management
P.O. Box 4120 | 39016 Magdeburg | Germany

Tel.: +49 (0) 3 91 / 67-1 85 84
Fax: +49 (0) 3 91 / 67-1 21 20

www.fww.ovgu.de/femm

ISSN 1615-4274