INFORMATIK
BERICHTE

370 – 08/2014

Metaheuristics for the Vehicle Routing Problem with Clustered Backhauls and 3D Loading Constraints

Andreas Bortfeldt, Thomas Hahn, Dirk Männel, Lars Mönch

Fakultät für Mathematik und Informatik
D-58084 Hagen

Metaheuristics for the Vehicle Routing Problem with Clustered Backhauls and 3D Loading Constraints

Andreas Bortfeldt[1*], Thomas Hahn[2], Dirk Männel[1], Lars Mönch[2]
[1] Otto von Guericke University, Universitätsplatz 2, 39106 Magdeburg, Germany
andreas.bortfeldt@ovgu.de, dirk.maennel@gmx.de
[2] University of Hagen, Universitätsstraße 1, 58097 Hagen, Germany
{thomas.hahn1|lars.moench@fernuni-hagen.de}
[*] Corresponding author, phone: 0049 391 67 11 8 42

Abstract

In this paper, we extend the vehicle routing problem with clustered backhauls (VRPCB) to an integrated routing and three-dimensional loading problem, called VRPCB with 3D loading constraints (3L-VRPCB). In the VRPCB each customer is either a linehaul or a backhaul customer and in each route all linehaul customers must be visited before any backhaul customer. In the 3L-VRPCB, each customer demand is given as a set of 3D rectangular items (boxes) and the vehicle capacity is replaced by a 3D loading space. Moreover, some packing constraints, e.g. concerning stacking of boxes, are also integrated. A set of routes of minimum total length has to be determined such that each customer is visited once. For each route two packing plans have to be provided that stow all boxes of all visited linehaul and backhaul customers, respectively, taking into account the additional packing constraints. We propose two hybrid algorithms for solving the 3L-VRPCB, each of them consisting of a routing and a packing procedure. The routing procedures follow different metaheuristic strategies (Large vs. Variable Neighborhood Search) and in both algorithms a Tree Search heuristic is responsible for packing boxes. Extensive computational experiments were carried out using 95 3L-VRPCB benchmark instances that were derived from well-known VRPCB instances. Good results are also achieved for the capacitated vehicle routing problem with 3D loading constraints as a special case of the 3L-VRPCB.

Key words: Transportation, vehicle routing with clustered backhauls, packing, integrated routing and packing problem, large neighborhood search, variable neighborhood search, tree search.

1 Introduction

Integrated vehicle routing and loading problems arise in transportation logistics if companies are interested in optimizing both the routing of vehicles and the corresponding loading of goods. In 2006, Gendreau et al. first formulated and solved an integrated routing and loading problem, namely the capacitated vehicle routing problem (CVRP) with three-dimensional (3D) loading constraints (abbreviated as 3L-CVRP). In contrast to the classical CVRP, customer demands are represented as sets of parallelepipeds (called boxes) and the scalar capacity of a vehicle is replaced by a 3D rectangular loading space. Moreover, several packing constraints often occurring in real-world settings, e.g. concerning the stacking of goods, are taken into account. Meanwhile, other solution methods have been proposed for the 3L-CVRP as well as for the related routing and loading problem with time windows (see Section 2). However, not all basic routing problems (cf. Toth and Vigo, 2002) have been combined with a 3D loading problem so far.

One of them is the vehicle routing problem with backhauls (VRPB) that includes two types of customers, namely linehaul customers that receive goods from a central depot and backhaul customers that send goods back to the depot. We focus on the VRPB variant with clustered backhauls (cf. Parragh et al., 2008). Each customer is either a linehaul customer or a backhaul customer. One has to determine a set of routes for a given number of homogeneous vehicles such that each customer demand is satisfied and the total traveled distance is minimized. The routes have to satisfy the following constraints:

(1) Each customer is visited exactly one time.

(2) Each route starts and ends at the central depot.

(3) Each route contains at least one linehaul customer.

(4) In each route all linehaul customers are visited before the backhaul customers (if any).

(5) The capacity of the vehicles has to be taken into account, i.e., neither the sum of the (scalar) demands of the linehaul customers nor the sum of the demands of the backhaul customers can exceed the vehicle capacity.

(6) Each vehicle performs just one trip, i.e. the number of routes is given by the number of vehicles.

In this paper, we extend the vehicle routing problem with clustered backhauls (VRPCB) to an inte-

grated routing and loading problem, called VRPCB with 3D loading constraints (3L-VRPCB). Applications of the VRPCB can be found, e.g. in the distribution of groceries and in the handling of returnable bottles (cf. Ropke and Pisinger 2006b). Extending the VRPCB to an integrated routing and loading problem leads to a more realistic modeling for practical routing scenarios with backhauls.

Our approach is similar to Gendreau et al. (2006); in particular the VRPCB is combined with the same packing constraints as the CVRP. We then propose two hybrid algorithms for the 3L-VRPCB each consisting of two separate procedures for routing and packing. For the first hybrid algorithm the routing procedure by Ropke and Pisinger (2006a) for the Pickup and Delivery Problem with Time Windows (PDPTW) has been adopted that is based on the Adaptive Large Neighborhood Search (ALNS) metaheuristic. For the second hybrid algorithm a Variable Neighborhood Search (VNS) routing procedure has been developed. Both hybrid algorithms integrate the 3D packing procedure by Bortfeldt (2012) that performs an incomplete tree search (TRS).

Extensive computational experiments have been conducted by means of 95 new 3L-VRPCB benchmark instances that were derived from the well-known VRPCB instances by Gottschalckx and Jacobs-Blecha (1989) and by Toth and Vigo (1997), respectively.

The remainder of the paper is organized as follows. In Section 2, the relevant literature is discussed. The 3L-VRPCB is formulated in Section 3. The proposed hybrid algorithms are described in Section 4. Section 5 is dedicated to the computational experiments. First, new benchmark instances for the 3L-VRPCB are introduced and the hybrid algorithms are calibrated. Then the numerical results are presented and analyzed. In Section 6, some conclusions are drawn and future research directions are proposed.

2 Related work

In our literature review, we will focus on the vehicle routing problem with clustered backhauls and on integrated vehicle routing and 3D loading problems. We refer the reader to Toth and Vigo (2002) and Golden et al. (2008) for a comprehensive survey on vehicle routing. Different types of vehicle routing problems with backhauls are surveyed by Parragh et al. (2008).

2.1 Exact approaches for the VRPCB

Exact approaches for the VRPCB were published, e.g., by Toth and Vigo (1997) and by Mingozzi et al. (1999). In the first paper, an integer programming formulation is proposed. Based on relaxations, lower bounds are determined. These bounds are used within a branch and bound approach to determine optimal solutions. A binary program is proposed in the second paper. The main idea of this exact approach consists in reducing the number of binary variables by considering the dual of the LP relaxation of the exact formulation. Both methods proved to be able to solve instances with up to 100 customers.

2.2 Conventional heuristic approaches for the VRPCB

Next, we discuss conventional heuristic approaches. An early constructive heuristic was derived from the Savings Algorithm (cf. Clarke and Wright, 1964) by Deif and Bodin (1984). Goetschalckx and Jacobs-Blecha (1989) propose a heuristic approach based on space-filling curves. Linehaul and backhaul customers are transformed from points in the plane into points along a line using the space-filling curve transformation. These two point sets are used to determine feasible routes. Then each linehaul route is merged with the nearest backhaul route with respect to the space-filling mapping. Toth and Vigo (1996) suggest a cluster-first, route-second algorithm for both the VRPCB and the related asymmetric problem (AVRPCB). A cluster is a group of customers that contains only linehaul or backhaul customers. Thangiah et al. (1997) propose a two-phase approach for the VRPBC with time windows. First, an initial solution is determined using an insertion heuristic. Then, in a second phase, a λ interchange and a 2-opt* procedure is applied to improve the initial solution.

2.3 Metaheuristic approaches for the VRPCB

Several solution methods to tackle the VRPCB based on metaheuristics can be found in the literature. Osman and Wassan (2002) suggest a reactive tabu search heuristic. Brandao (2006) presents a tabu search based procedure. Tavakkoli-Moghaddam et al. (2006) and Saremi et al. (2007) publish memetic

algorithms. Neural network based approaches are presented in Ghaziri and Osman (2006). A reactive tabu adaptive memory programming search is discussed by Wassan (2007). Ropke and Pisinger (2006b) demonstrate that the VRPCB and other backhaul problems (with and without time windows) can be tackled in a unified way by modeling them as rich pickup and delivery problem with time windows (Rich PDPTW). The required problem transformation is based on introducing appropriate precedence constraints. Ropke and Pisinger (2006b) propose a ALNS heuristic for solving the Rich PDPTW that achieves an excellent solution quality for the VRPCB and other backhaul problems. A multi-ant colony system is published by Gajpal and Abad (2009). An iterated local search heuristic yielding high-quality solutions is proposed by Arráiz and Palhazi Cuervo (2011). In their heuristic the search is not restricted to the space of feasible solutions; instead, solutions are considered temporarily that do not satisfy the capacity constraint. Zachariadis and Kiranoudis (2012) propose an effective local search heuristic which explores rich neighborhoods composed of exchanges of variable-length customer sequences. Vidal et al. (2014) suggest a unified solution framework for a great variety of multi-attribute vehicle routing problems including the VRPCB. While VNS is successfully applied for a number of different VRPs (cf. Kytöjoki et al. 2007, Hemmelmayr et al. 2009, Kritzinger et al. 2011), to the best of our knowledge, VNS approaches have not been used up to now to solve the VRPCB.

2.4 Solution approaches for integrated routing and 3D loading problems

Iori and Martello (2010) survey the state of the art in the field of integrated vehicle routing and loading problems. Generally, the literature is still limited and this applies in particular to the 3D case.

The 3L-CVRP was introduced by Gendreau et al. (2006) with five additional packing constraints frequently occurring in freight transportation. These include a weight constraint, a last-in-first-out (LIFO) loading constraint, an orientation constraint, a support constraint, and a stacking constraint (see Section 3 for details). Gendreau et al. suggest a two-stage tabu search algorithm for solving the 3L-CVRP. The "outer" tabu search serves for planning the routes, while the "inner" tabu search solves a 3D strip packing problem in order to load a vehicle according to a given customer sequence. Tarantilis et al. (2009) propose a hybrid procedure combining the strategies tabu search and guided local search. They use a collection of plain packing heuristics. Fuellerer et al. (2010) develop an ant colony algorithm for routing that is integrated with fast but effective packing heuristics. Wang et al. (2010) design a two-phase tabu search algorithm for routing that cooperates with two constructive packing heuristics. This hybrid algorithm was further developed by Zhu et al. (2012). Wisniewski et al. (2011) propose a tabu search for routing and a randomized bottom left-based packing algorithm. Bortfeldt (2012) suggests a hybrid algorithm for the 3L-CVRP with a tabu search procedure for routing and a tree search algorithm for loading vehicles. Ruan et al. (2013) present a honey bee mating algorithm for routing that is combined with six loading heuristics. Finally, Lacomme et al. (2013) propose an effective hybrid procedure for the 3L-CVRP that, however, does not consider all 3D packing constraints introduced by Gendreau et al. (2006).

Moura and Oliveira (2009) specify the VRP with time windows and 3D loading constraints (3L-VRPTW) with two objectives and present two heuristic procedures for this problem. The number of vehicles is minimized with higher priority, whereas the total travel distance is minimized with lower priority. The authors do not consider the weight and the stacking constraint of the 3L-CVRP while the other packing constraints (see above) are adopted. Another hybrid algorithm for solving the 3L-VRPTW was recently suggested by Bortfeldt and Homberger (2013). It consists of an evolutionary strategy and two tabu search procedures. Meanwhile, there are several solution procedures for the 3L-CVRP and also some for the 3L-VRPTW. However, algorithms for other integrated routing and 3D loading problems were not published so far.

A hybrid algorithm for 3L-VRPCB is sketched in Bortfeldt et al. (2013). However, this extended abstract is considerably enhanced by the present paper. We propose a second hybrid approach based on VNS and include much more computational results.

3 Problem formulation

In this section, we describe the 3L-VRPCB more formally. We assume that l linehaul customers (indexed by 1,...,l), b backhaul customers (indexed by l+1,....,l+b), and a single depot (with index 0) are given. Each customer has a set I_i of boxes with known dimensions that are either to be transported from the depot to the customer (i = 1,...,l) or from the customer to the depot (i = l+1,...,l+b). The set I_i

includes m_i rectangular packing pieces (boxes) I_{ik} ($k = 1,...,m_i$) and the box I_{ik} has the length l_{ik}, the width w_{ik}, and the height h_{ik} ($i = 1,...,l+b$, $k = 1,...,m_i$).

Let $V = \{0,1,..,l,l+1,\ldots,l+b\}$ denote the set of all customer sites (also called nodes) including the depot site. Let E be a set of undirected edges (i,j) connecting all node pairs ($0 \le i < j \le l+b$) and let $G = (V,E)$ be the resulting graph. Let a distance c_{ij} ($c_{ij} \ge 0$) be assigned to each edge (i,j) ($0 \le i < j \le l+b$). Finally, there are v_{max} identical vehicles with a rectangular loading space with length L, width W, and height H. Each vehicle is rear-loaded.

The loading space of each vehicle is embedded in the first octant of a Cartesian coordinate system in such a way that the length, width and height of the loading space lie parallel to the x, y, and z axes. The placement of a box I_{ik} in a loading space is given by the coordinates x_{ik}, $y_{ik,}$, and z_{ik} of the corner of the box that is closest to the origin of the coordinates system; in addition, an orientation index o_{ik} indicates which of the possible spatial orientations is selected ($i = 1,...,l+b$, $k = 1,...,m_i$). A spatial orientation of a box is given by a one-to-one mapping of the three box dimensions and the three coordinate directions.

A packing plan P for a loading space comprises one or more placements and is regarded as feasible if the following three conditions hold: (FP1) each placed box lies completely within the loading space; (FP2) any two boxes that are placed in the same truck loading space do not overlap; (FP3) each placed box lies parallel to the surface areas of the loading space. Figure 1 shows a loading space with placed boxes.

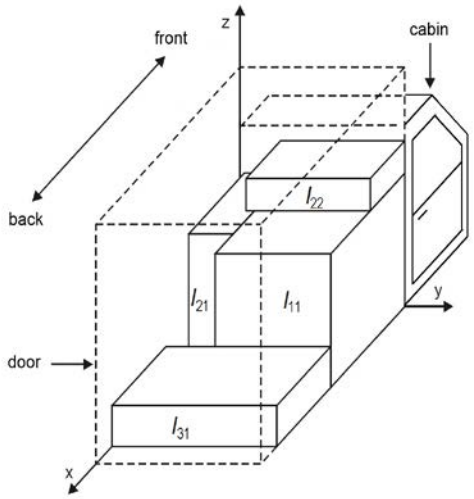

Figure 1: A loading space with placed boxes.

A route R is a sequence $(0,c_1,...,c_n,0)$ that starts and ends with the depot and includes $n \ge 1$ pairwise different customers ($c_i > 0$). A route is feasible if it includes at least one linehaul customer ($1 \le c_i \le l$) and all linehaul customers precede the first backhaul customer ($l+1 \le c_i \le l+b$) if any.

A solution of the 3L-VRPCB is a set of v ($v \ge 1$) triples $(R_\mu, P_{\mu,l}, P_{\mu,b})$, where R_μ is a route and $P_{\mu,l}$ and $P_{\mu,b}$ denote two packing plans ($\mu = 1,...,v$). A solution is called feasible if the following conditions are satisfied:

(F1) all routes R_μ and all packing plans $P_{\mu,l}$, $P_{\mu,b}$ are feasible ($\mu = 1,...,v$);

(F2) each customer appears exactly in one route;

(F3) the packing plan $P_{\mu,l}$ includes the placements for all boxes of all linehaul customers of route R_μ, and $P_{\mu,b}$ comprises the placements for all boxes of all backhaul customers of route R_μ (if any) ($\mu = 1,...,v$);

(F4) the number of routes v does not exceed the given number of vehicles v_{max}.

In addition, the following packing constraints are integrated:

(C1) **Weight constraint:** Each box set I_i has a positive weight d_i ($i = 1,...,l+b$) and the total weight of all boxes in a packing plan $P_{\mu,l}$ or $P_{\mu,b}$ cannot exceed a maximum load weight D ($\mu = 1,...,v$).

(C2) **LIFO constraint:**

Linehaul part: Let c and c' be two linehaul customers and c is visited before c' in route μ ($\mu \in \{1,...,v\}$); let b and b' two boxes that belong to c and c', respectively. Then b' cannot be placed in packing plan $P_{\mu,l}$ between b and the rear of the vehicle or above b.

Backhaul part: Let c and c' be two backhaul customers and c is visited before c' in route μ ($\mu \in \{1,...,v\}$); let b and b' two boxes that belong to c and c', respectively. Then b cannot be placed in packing plan $P_{\mu,b}$ between b' and the rear of the vehicle or above b'. By these constraints it is ensured that all boxes of each linehaul customer can be unloaded by pure shifts in x-direction without moving other (linehaul) boxes. Likewise it is ensured that all boxes of each backhaul customer can be loaded by pure shifts in negative x-direction without moving other (backhaul) boxes.

(C3) **Orientation constraint:** The height dimension of all boxes is fixed while horizontal 90° turns of boxes are allowed. Thus only two of six values are allowed for the orientation index o_{ik} of a placement (i = 1,..., l+b; k = 1,...,m_i).

(C4) **Support constraint:** If a box is not placed on the floor, a certain percentage a of its base area has to be supported by other boxes.

(C5) **Stacking constraint:** A fragility attribute f_{ik} (i = 1,...,l+b, k = 1,...,m_i) is assigned to each box. If a box is fragile (f_{ik} = 1) only other fragile boxes may be placed on its top surface, whereas both fragile and non-fragile boxes may be stacked on a non-fragile box (f_{ik} = 0).

Finally, the 3L-VRPCB consists of determining a feasible solution that meets the constraints (C1) - (C5) and minimizes the total travel distance of all routes.

In contrast to the VRPCB, it is now allowed that solutions contain less than v_{max} routes and we consider this a more realistic setting. As mentioned earlier, the added packing constraints (C1) - (C5) were just adopted from the 3L-CVRP. One advantage of this procedure is that computational results for the 3L-CVRP and the 3L-VRPCB, for example, mean volume utilizations, can be easier compared.

4 Two hybrid metaheuristic algorithms

The 3L-VRPCB is NP-hard since it contains the NP-hard CVRP (cf., e.g., Toth and Vigo 2002) as a special case. Moreover, as an integrated routing and loading problem the 3L-VRPCB is extremely difficult to solve (cf. Iori and Martello, 2011, p. 8). Thus the application of metaheuristic search strategies (including heuristic graph search) seems to be the only viable option to develop solution methods that provide high-quality solutions for large-size instances within acceptable computing time.

In the sequel, we describe two hybrid metaheuristic algorithms for the 3L-VRPCB, each consisting of two separate procedures for routing and packing. The ALNS heuristic for solving the PDPTW by Ropke and Pisinger (2006a) serves basically as routing procedure for the first 3L-VRPCB algorithm. The routing procedure of the second 3L-VRPCB algorithm is a VNS heuristic. We are interested in answering the question whether the results of the hybrid approaches are influenced by the neighborhood structures used in the routing procedure or not. The neighborhood structures applied by ALNS tend to change a larger portion of a current solution, while the neighborhood structures used in VNS typically change a much smaller part of a current solution.

Both 3L-VRPCB algorithms integrate the 3D packing algorithm by Bortfeldt (2012) that is based on a tree search approach. The routing procedure is the main (first called) module and the packing procedure is the subordinated module in both 3L-VRPCB algorithms.

Subsequently, the individual procedures are specified; some details of already known algorithms are omitted and the reader is referred to the mentioned original papers for a complete description.

4.1 First routing procedure

The first routing procedure is based on the ALNS heuristic by Ropke and Pisinger (2006a). Generally, we try to keep our routing procedure close to the original ALNS heuristic. However, some parts of the original routing procedure are omitted to achieve a slightly simpler algorithm while two operators are added or modified on the other hand. Of course, the routing procedure for solving the 3L-VRPCB has to be integrated with the packing module mentioned above and emphasis is laid on this part in the following description.

The routing procedure is roughly outlined in Figure 2. First, an initial solution is constructed.

Afterwards, an iterative neighborhood search is carried out until a computing time limit is exceeded.

```
3l_vrpcb_alns (in: problem data, parameters, out: best solution s_best)
    construct initial solution s_curr and set s_best := s_curr
    while stopping criterion is not met do
        select number of customers to be removed ξ
        select removal heuristic Rh and insertion heuristic Ih
        determine next solution: s_next := Ih(Rh(s_curr, ξ))
        check acceptance of s_next
        if s_next is accepted then
            s_curr := s_next
            if (f(s_curr) < f(s_best)) then s_best := s_curr endif
        endif
    endwhile
end.
```

Figure 2: ALNS-based routing algorithm for the 3L-VRPCB.

Within each iteration, a number ξ of customers to be removed is selected randomly from the set $\{r_{min},\ldots,r_{max}\}$. Several removal and insertion heuristics are available. Among them one removal and one insertion heuristic are selected randomly per iteration. The selection probabilities for the removal and insertion heuristics depend on their relative success in the previous search and are dynamically updated by a so-called learning layer as in the original ALNS. The next solution is generated by the selected heuristics *Rh* and *Ih* according to $s_{next} := Ih(Rh(s_{curr}, \xi))$. If s_{next} is accepted in a dedicated test it becomes the new current solution s_{curr} and the best solution s_{best} is updated if necessary. Otherwise, the initial solution of the next iteration s_{curr} remains unchanged. As in the original ALNS the acceptance test follows the well-known simulated annealing rule and according to this the search is embedded in an annealing process with a geometric cooling schedule. Differently to the original ALNS no noise term is applied to the objective function.

Since the number of vehicles is limited it may happen that the initial solution or a later generated solution is incomplete, i.e. some customers are missing. To cope with this situation the concept of a virtual request bank is used as in the original ALNS (see Ropke and Pisinger 2006a, p. 2). The objective function is specified as the sum $f(s) = ttd(s) + M \cdot nmc(s)$, where s is a given solution, *ttd* stands for its total travel distance, *nmc* is the number of missing customers and M is a sufficiently large constant. By this definition solutions with less missing customers are always preferred.

In the following, some parts of the ALNS heuristic are explained in more detail.

4.1.1 Removal and insertion heuristics

The removal and insertion heuristics are briefly summarized in Table 1. They are basically adopted from the original ALNS heuristic by Ropke and Pisinger (2006a). However, the more difficult insertion heuristics Regret-m (m > 3) of the original ALNS are omitted here.

Table 1: Removal and insertion heuristics of the LNS heuristic for 3L-VRPCB.

Heuristic	Description
Random removal Rh_R	Removes iteratively customers that are selected at random.
Shaw removal Rh_S	Removes iteratively customers that are related in terms of location and weight.
Worst removal Rh_W	Removes iteratively a customer whose removal leads to the largest cost (total travel distance) reduction.
Tour removal Rh_T	Removes all customers from a randomly chosen route. If less than ξ customers are removed in this way, further customers will be removed with Shaw removal.
Greedy insertion Ih_G	Inserts iteratively customers into the solution such that the increase of the cost function is minimal.
Regret-2 insertion Ih_{R2}	Inserts iteratively customers into the solution such that the gap in the cost function between inserting the customer into its best and its second best route is maximal.
Regret-3 insertion Ih_{R3}	Inserts iteratively customers into the solution such that the sum of two gaps in the cost function is maximal. The first gap results from inserting the customer into its best and its second best route, while the second gap results from inserting the customer into its best and its third best route.

Within the Shaw removal the relatedness of customers is expressed by means of two factors, namely the location of the customers and the weights of their item sets. Hence, the relatedness of the two customers i and j is calculated by the blended index $r(i,j) = w_{r1}c'_{ij} + w_{r2}|v'_i - v'_j|$, $1 \le i < j \le l+b$, where

c'_{ij} denotes the normalized distance between i and j and lies in the interval [0,1]; v'_i denotes the normalized weights of customer i where the normalization interval is [0,1] for linehaul customers and [-1,0] for backhaul customers, respectively. The weights w_{rp} (p = 1, 2) allow for a different weighting of the distance and weights difference.

Finally, the Tour removal has been added in order to drive the search into regions where feasible solutions with less tours can be found. Although the minimization of the number of tours is not an explicit goal this procedure can be helpful to identify high-quality solutions in terms of total travel distance.

The initial solution is constructed by means of the Regret-2 insertion heuristic (see Subsection 4.1.2 for details) starting with an empty solution.

4.1.2 Integration of routing and packing

For each route of a solution it has to be ensured that the boxes of all linehaul customers can be stowed together in a vehicle's loading space. Likewise it has to be guaranteed that all backhaul boxes can be packed in a loading space. The integration of routing and packing is guided by the following principles:

- Consider a route with two feasible packing plans for the linehaul boxes and the backhaul boxes. If some of the linehaul customers are removed then it will be mostly possible to store the boxes of the remaining linehaul customers in a feasible way, too. (Exceptions of this rule are discussed at the very end of this subsection.) However, if one or more new linehaul customers are inserted in the route then it might become impossible to store all appropriate boxes in a feasible packing plan. The same applies for the backhaul part of the route (if any). Therefore, packing checks are integrated in the insertion heuristics exclusively.
- Generally, multiple customers are inserted per route if a new solution is completed by means of an insertion heuristic. A packing check is made each time when another customer is inserted into a route. Of course, if a linehaul customer was inserted only the linehaul part of the route is checked and only a packing check of the backhaul part is made in the opposite case.
- In each packing check of the linehaul or backhaul part of a route it is checked whether all boxes of all relevant (linehaul or backhaul) customers can be packed together in one loading space in such a way that the basic conditions (FP1) – (FP3) and all packing constraints (C1) – (C5) are observed.
- During the entire search, a packing check only returns whether the boxes of a given set can be packed together in a feasible way while no packing plans are returned. At the end of the search all routes of the best found solution are checked in terms of packing once again and for each route feasible packing plans for the linehaul and the backhaul part are now provided.

Below two of the insertion heuristics, namely the Greedy insertion (Figure 3) and the Regret-2 insertion (Figure 4) are described in more detail. Both the insertion heuristics are based on the procedure select_best_insertions that performs all packing checks and is shown in Figure 5. Note that the Regret-3 insertion works similar to the Regret-2 insertion.

The Greedy insertion heuristic takes an incomplete solution, a set of missing customers and the best solution so far as input values. In each loop cycle the best (minimum cost) insertion is determined related to the customers still missing and implemented before the set of missing customers is updated. The procedure select_best_insertions is used to deliver the best insertion for a given customer. In each cycle the number of still missing customers nmc_{wi} is counted for which no feasible insertion was found at all. If in any cycle nmc_{wi} is greater than the number of missing customers $nmc(s_{best})$ in the best solution found so far then a further computation is useless and the heuristic will return. Otherwise a solution is provided in the end that has no more missing customers than the best solution so far or is even a feasible and complete solution.

The Regret-2 insertion heuristic starts with the same input values as the Greedy insertion heuristic. In each cycle, the insertion with maximum regret value is determined related to the customers still missing and implemented before the set of missing customers is updated. The best two insertions per missing customer belonging to different routes are sought by procedure select_best_insertions. The regret value of the best insertion of a customer is given by the cost difference between the second best and the best insertion; the insertion with maximum regret value is then carried out per cycle.

```
greedy_insertion (in: set of missing customers Cm, s_best, inout: solution s)
    repeat
        min_cost := ∞
        nmc_wi := 0                                   // no. missing customers without insertion
        for all c ∈ Cm do
            I_best(c) := select_best_insertions(s,c,1)   // set I_best(c) receives best c-insertion
            if |I_best(c)| = 0 then nmc_wi := nmc_wi + 1   // customer without insertion
            else
                if cost of best c-insertion < min_cost then
                    c_ins := c; min_cost := cost of best c-insertion endif
            endif
        endfor
        if nmc_wi > nmc(s_best) then return endif          // no useful solution
        if nmc_wi <|Cm| then update s by insertion in I_best(c_ins); Cm := Cm \ {c_ins} endif
    until (|Cm| = 0 or nmc_wi =|Cm|)
end.
```

Figure 3: Greedy insertion heuristic.

If there is only one feasible insertion for a customer, an infinite regret value is used to force the heuristic to select this insertion. If no feasible insertion can be found at all for one or more customers within a cycle the Regret-2 insertion proceeds in a similar manner as Greedy insertion. Again, in case of success a solution is provided that is as least as good as the best solution so far in terms of missing customers or is even a feasible and complete solution.

```
regret_2_insertion (in: set of missing customers Cm, s_best, inout: solution s)
    repeat
        max_regret_value := 0
        min_cost := ∞
        nmc_wi := 0                                   // no. missing customers without insertion
        for all c ∈ Cm do
            I_best(c) := select_best_insertions(s,c,2)   // set I_best(c) receives best two c-insertions
            if |I_best(c)| = 0 then nmc_wi := nmc_wi + 1  // customer without insertion
            else
                if |I_best(c)| = 1 then regret_value := ∞
                else regret_value := cost difference of best two c-insertions endif
                if regret_value > max_regret_value or
                   regret_value = max_regret_value and cost of best c-insertion < min_cost then
                    c_ins := c; max_regret_value := regret_value; min_cost := cost of best c-insertion
                endif
            endif
        endfor
        if nmc_wi > nmc(s_best) then return endif          // no useful solution
        if nmc_wi < |Cm|then update s by best insertion in I_best(c_ins); Cm := Cm \ {c_ins} endif
    until (|Cm| = 0 or nmc_wi = |Cm|)
end.
```

Figure 4: Regret-2 insertion heuristic.

The procedure select_best_insertions is organized in two parts. In the first part (*for*-loop) all potential insertions of a given customer *c* into any route of a given solution *s* are provided. Each insertion must be feasible (only) in terms of weight (constraint (C1)) and volume (the total volume of the goods must not exceed the loading space volume). The minimum cost insertions of all routes are collected in a list I_{cand}. In the second part (*while*-loop), the insertions of I_{cand} are examined by ascending costs. In each cycle the currently minimum cost insertion ins_{best} undergoes a 3D packing check (including constraints (C2) to (C5)). If the outcome is positive insertion ins_{best} is included into the set of best insertions I_{best} (and removed in I_{cand}). Otherwise the next cheapest insertion for the route of ins_{best} (if any) will replace ins_{best} in list I_{cand}. As stated above, either the linehaul part or the backhaul part of a route is checked (in both parts of the procedure) depending on the type of the customer *c*. The procedure returns if I_{best} has enough (n_{ins}) insertions or if I_{cand} is empty. Any two insertions in I_{best} belong to different routes.

Two features of the procedure select_best_insertions should be stressed. First, the one-dimensional checks are made before 3D packing checks are carried out. Second, all possible insertions are first evaluated and sorted by cost *before* the 'expensive' packing checks are made. By this practice, called "evaluating first, packing second", the packing effort is kept low since the packing checks can be aborted each time after few (3D-) feasible insertions have been detected.

```
select_best_insertions (in: solution s, customer c, no. of required insertions n_ins,
                        out: set of best c-insertions I_best)
    I_best := ∅; list of insertion candidates I_cand := ∅
    for all routes r of solution s do
        I_route(r) := set of all 1D-feasible insertions of c in route r
        sort I_route(r) by ascending cost
        if |I_route(r)| > 0 then I_cand := I_cand ∪ {I_route(r)(1)} endif    // add first insertion of I_route(r)
    endfor
    while |I_best| < n_ins and |I_cand| > 0 do
        sort I_cand by ascending cost
        best insertion ins_best := I_cand(1); I_cand := I_cand \ {ins_best}
        if 3D packing check of s and ins_best successful then
            I_best := I_best ∪ {ins_best}   // next best insertion found
        else  r := route of ins_best
            I_route(r) := I_route(r) \ {ins_best} // remove ins_best
            if |I_route(r)| > 0 then I_cand := I_cand ∪ {I_route(r)(1)} endif // add (new) first insertion
        endif
    endwhile
end.
```

Figure 5: Procedure select_best_insertions with packing check.

Sometimes the boxes of a given set can be stored in the loading space in a feasible way while this is no longer the case after some of the boxes were removed. Such a situation may occur, e.g., if a box was removed that is needed to provide sufficient support for another fragile box. Since such cases occur very rarely a packing check is not carried out after a box arrangement in a loading space was only reduced (while no other box was added). However, if a solution turns out to be a new best solution all its routes undergo an additional packing check. In particular all routes are checked that did result earlier by a pure removing of items. By this measure it is prevented that the best solution ever includes an infeasible route in terms of packing.

Our implementation of the ALNS routing procedure can also be applied to the (1D) VRPCB. In this situation, the 3D packing test is omitted and only the packing constraint (C1) is checked.

4.2 Second routing procedure

VNS is a local search-based metaheuristic introduced by Mladenovic and Hansen (1997) and Hansen and Mladenovic (2001). The main idea is to enrich a simple local search method to enable it to escape from local optima. This is carried out by restarting the local search from a randomly chosen neighbor of the incumbent solution. This restarting step, so-called shaking, is performed using different neighborhoods of increasing sizes.

There are many different variants of VNS, such as variable neighborhood descent (VND) or re-

duced variable neighborhood search (RVNS). Our approach will be based on the basic VNS. The overall procedure for 3L-VRPCB is shown in Figure 6. Here, we denote by N_k the kth neighborhood. We describe the main ingredients of our VNS approach shown in Figure 6 in more detail in the remainder of this subsection.

4.2.1 Neighborhood structures

Our VNS algorithm designed for the 3L-VRPCB operates on the final solution representation, i.e., each customer is assigned to exactly one route and each route is assigned to a certain vehicle. We start by describing the different neighborhood structures. We use similar neighborhood structures as in Hemmelmayr et al. (2009) for the periodic VRP. However, in our setting, the neighborhood structures are separately applied to the linehaul and the backhaul part of the routes because linehaul and backhaul customers are clustered. The neighborhood structures used are based on the move and the cross-exchange operator.

```
3l_vrpb_vns (in: problem data, parameters, out: best solution s_best)
    select K different neighborhood structures N_k
    determine an initial solution s_curr and set s_best := s_curr
    set k := 1
    while stopping criterion is not met do
        Shaking: choose randomly s_next ∈ N_k(s_curr)
        Local search: improve s_next by a local search method
        if f(s_next) < f(s_best) and the packing test for s_next is successful then
          k=1
          s_curr := s_next; s_best := s_curr
        else
          k=(k mod K)+1
        endif
    endwhile
end.
```

Figure 6: VNS-based Routing algorithm for the 3L-VRPCB.

The move operator removes a certain segment of the linehaul or backhaul part of a randomly selected route and inserts it into the linehaul or backhaul part of a second randomly chosen route, respectively. The starting position in each of the corresponding parts of the two routes is randomly selected. Segments up to three customers are considered by the move operator in the researched problem. We abbreviate the move operator for segments of $\min(k,l)$ or $\min(k,b)$ customers by MoveLinehaul(k), MoveBackhaul(k), respectively. The orientation of the segments and routes is not changed when the move operator is applied.

The cross-exchange operator swaps two segments across the linehaul or backhaul part of two different randomly selected routes. Segments that contain up to six customers are considered. The starting position for each of the segments is randomly chosen. The cross-exchange operators will be abbreviated for segments of $\min(k,l)$ or $\min(k,b)$ customers by CrossExchangeLinehaul(k) and CrossExchangeBackhaul(k), respectively. The orientation of the segments and routes again is not changed when the cross-exchange operator is applied.

The set of neighborhood structures is the main ingredient of the shaking phase of the VNS approach. Based on some preliminary computational experiments, we found that the sequence of neighborhood structures shown in Table 2 performs well. The size of the neighborhood structures is determined by the length of the customer segments in the linehaul or backhaul part of route that is a parameter of the neighborhood structure and the number of linehaul or backhaul customers. Note that the operations performed by the move and cross-exchange operators are different and one is not a special case of the other. Any combined neighborhood structure containing move and cross-exchange operations is not nested. Also note that the sizes of the neighborhood structures depicted in Table 2 are not increasing, i.e., when switching from N_3 to N_4 as well as N_6 to N_7 and N_{12} to N_{13} the neighborhood sizes decrease.

4.2.2 Initial solution

The initial solution is determined by a constructive heuristic that is derived from the Savings Algorithm for the CVRP (cf. Clarke and Wright, 1964). At first, separate solutions for the linehaul and for the backhaul customers, respectively, are constructed by a Savings heuristic that generates the solution route by route. Of course, two provisional linehaul or backhaul routes are only linked if the outcome of the weight test *and* the packing test is positive. It is ensured that the number of routes for backhaul customers does not exceed the number of routes for linehaul customers. At last, linehaul and backhaul routes are pairwise amalgamated following the Savings principle.

4.2.3 Local search

The local search procedure is applied to each linehaul part of a route that is changed during the shaking phase. If the linehaul part of the route is changed then the corresponding backhaul route is also considered simultaneously in the local search procedure since the visit of the backhaul customers is affected by changes of the linehaul customers. The local search procedure is based on 2-opt and 3- opt (Lin 1965). The 2-opt or 3-opt procedure is based on the idea to delete 2 and 3 edges, respectively between customers on a route and then reconnecting the customers in all possible ways. It consists of two phases. In a first phase, 2-opt is applied. We then apply 3-opt without sequence inversion in the second phase.

Table 2: Sequence of the neighborhood structures for $K = 18$.

k	Operator	Maximal number of customers
1	MoveLinehaul(1)	1
2	MoveLinehaul(2)	$\min(2,l)$
3	MoveLinehaul(3)	$\min(3,l)$
4	MoveBackhaul(1)	1
5	MoveBackhaul(2)	$\min(2,b)$
6	MoveBackhaul(3)	$\min(3,b)$
7	CrossExchangeLinehaul(1)	1
8	CrossExchangeLinehaul(2)	$\min(2,l)$
9	CrossExchangeLinehaul(3)	$\min(3,l)$
10	CrossExchangeLinehaul(4)	$\min(4,l)$
11	CrossExchangeLinehaul(5)	$\min(5,l)$
12	CrossExchangeLinehaul(6)	$\min(6,l)$
13	CrossExchangeBackhaul(1)	1
14	CrossExchangeBackhaul(2)	$\min(2,b)$
15	CrossExchangeBackhaul(3)	$\min(3,b)$
16	CrossExchangeBackhaul(4)	$\min(4,b)$
17	CrossExchangeBackhaul(5)	$\min(5,b)$
18	CrossExchangeBackhaul(6)	$\min(6,b)$

4.2.4 Feasibility issues

Infeasibilities with respect to the weight constraints (C1) can arise during the application of the neighborhood search operators during the shaking phase. Note that infeasibilities cannot occur in the local search procedure. It turned out in the course of the computational experiments that we have to allow VNS to deal with infeasible solutions in the beginning of the search process to obtain high-quality solutions. Similar observations were reported by Arráiz and Palhazi Cuervo (2011) for the VRPCB. We use a penalty term approach in the objective function before the solution is assessed. Therefore, the weight exceed of a route is multiplied with a certain penalty coefficient. Starting from an initial value α, this coefficient increases by a step size ψ each time an infeasible solution is assessed until a given upper limit β is reached. This drives the search towards feasible solutions.

4.2.5 Integration of routing and packing

As shown in Figure 6, each time a smaller objective function value compared to the best solution is

observed, a packing test is performed. The solution s_{next} is accepted as new incumbent solution if it improves the best solution found so far and feasible packing plans, fulfilling the constraints (C2) to (C5), can be determined for the linehaul and backhaul parts of all routes of s_{next} by the packing procedure. Note that the weight constraint (C1) is already checked within the shaking phase, but violations of this constraint are allowed to a certain extent.

The proposed VNS procedure can also be applied to the (1D) VRPCB. Similar to the LNS application, the 3D packing test is not performed and only the (C1) packing constraint is considered.

4.3 Packing procedure

For a given sequence of linehaul or backhaul customers and the corresponding set of boxes, the packing procedure tries to determine a complete solution, i.e. a packing plan stowing all given boxes. A depth first search is carried out by means of the recursive procedure add_placement shown in Figure 7. A stowage plan *currentSolution* is transferred and then extended in different variants by one further box placement for each procedure call.

As the search is started the solution *currentSolution* is set empty, the set *freeBoxes* is filled by all boxes and the list *potentialPlacements* is filled by all feasible box placements in the lower left back corner of the loading space $L \times W \times H$ (cf. Figure 1).

The procedure add_placement checks first whether the current packing check can be aborted or not. This is done, i.e. all running instances of procedure add_placement are aborted, if *currentSolution* is a complete solution or if the number of calls of add_placement exceeds a given limit *maxApCalls*. The current instance of the procedure is aborted if there is at least one free box without a potential placement, i.e. if a complete solution can no longer be achieved.

Candidates for the next placement are selected from the list *potentialPlacements* and provided in the list *currentPlacements*. All these placements are then tried alternatively. For each placement, the current solution, the set of free boxes, and the list of potential placements are updated accordingly before the procedure add_placement is called again. To update the list *potentialPlacements,* all potential placements are removed that can no longer be implemented. Additional potential reference points for new potential placements are determined as extreme points (see Crainic et al. 2008).

```
add_placement (in: freeBoxes, potentialPlacements, inout: currentSolution)
if all boxes stowed in currentSolution or number of procedure calls > maxApCalls then
        abort packing check endif
// abort current instance
if there is at least one free box without placement in potentialPlacements then return endif
provide list currentPlacements with potential placements that are currently to be tried
for i := 1 to |currentPlacements| do
        currentSolution' :=  currentSolution ∪ { currentPlacements(i) }          // add placement to solution
        freeBoxes' := freeBoxes \ { currentPlacements(i).box }                   // update free boxes
        potentialPlacements' := update(potentialPlacements)                      // update potential placements
        add_placement (currentSolution', freeBoxes', potentialPlacements')       // recursive call
endfor
end.
```

Figure 7: Packing procedure add_placement.

The selection of placements currently to be tried among all potential placements is governed by two rules. On the one hand, it is ensured that a vehicle is loaded from the front to the back, from bottom to top with lower priority, and from left to right with lowest priority. Hence, placements with smaller x-coordinates of the reference corner are preferred etc. On the other hand, the selection is made taking into account the LIFO constraint. If a linehaul sequence is given, placements of boxes are preferred that belong to customers that have to be visited later. Therefore, their boxes have to be loaded earlier, i.e. nearer to the cabin. The opposite procedure is implemented for a backhaul sequence. The placement selection is controlled by the integer parameters *maxBoxRankDiff* and *maxRefPoints* where higher parameter values lead to a larger set of currently tried placements.

All the linehaul and backhaul customer sequences that have ever been checked are collected in a cache to further accelerate the search. Whenever a sequence is tested in terms of packing, it is first searched in the cache. To speed up this search, for each customer c a separate table is established keeping the positions of all stored sequences including customer c. Thus, a customer sequence is searched

by examining only the cache positions of its first customer. The packing algorithm is only called if the sequence was not found in the cache and the sequence is then inserted in the cache together with the result of the packing test. Multiple checks of same routes are avoided by this procedure.

5 Computational experiments

The computational experiments are organized in four parts. In the *first part* we want to check whether the 1D variants of the hybrid algorithms (denoted by 1D-ALNS and 1D-VNS) are on a par with the state of the art heuristics for the VRPCB.

In the *second part* the 3D variants of the hybrid algorithms are tested addressing several issues. First, we are interested in the increase of total travel distances if 1D problem instances are replaced by corresponding 3D instances (see 5.1). A further issue is the impact of the computational effort on the solution quality. Of course, we are also interested in the relative strength of both hybrid algorithms and want to know whether the algorithms behave in a similar manner. In some preliminary tests, it was observed that for both hybrid algorithms and very different calibrations of the packing procedure the time spent for packing always exceeds 97% of the entire computing time. Hence, performing experiments with a significantly higher or lower relative packing effort seems not to be a viable option. However, what can be changed is the specific effort of a *single* packing check. The algorithms can be run using a smaller number of more thorough packing checks or a larger number of less thorough packing checks, and this alternative is examined. Accordingly, two 3D variants are specified for both hybrid algorithms. In the first 3D variant, the packing procedure is calibrated with a relative *high* specific computational effort per packing check and the hybrid algorithms are denoted by 3D-ALNS-H and 3D-VNS-H, respectively. In the second 3D variant, the packing procedure is calibrated with a relative *low* specific computational effort per packing check and the hybrid algorithms are denoted by 3D-ALNS-L and 3D-VNS-L, respectively.

As the hybrid ALNS algorithm proves to be the stronger one only this algorithm is subject of further experiments. In the *third part* the impact of different components of the hybrid ALNS algorithm on the solution quality is studied by means of 3L-VRPCB instances.

As the 3L-CVRP is a special case of the 3L-VRPCB (having only linehaul customers) in the *fourth part* the ALNS hybrid algorithm is applied to well-known 3L-CVRP benchmark instances.

The packing procedure and the VNS approach are coded in the C++ programming language using Visual Studio 2012 Express, while the ALNS scheme is implemented using the Java programming language under Eclipse 3.5.2. Preliminary experiments (in which total run times were varied) demonstrated that the impact of the different developing environments is negligible. All the experiments have been conducted on a PC with Intel Core i7-2600 (3.4 GHz, 16 GB RAM). Afterwards the used benchmark instances are introduced and the parameter setting is specified before the computational results are presented and analyzed.

5.1 Benchmark instances

The one-dimensional variants of the hybrid algorithms are tested by means of two well-known sets of VRPCB instances introduced by Goetschalckx and Jacobs-Blecha (1989) and by Toth and Vigo (1997). The first set (abbreviated as GJB) consists of 62 problem instances while the second set (abbreviated as TV) includes 33 problem instances. The ALNS hybrid algorithm is tested as 3L-CVRP solution method using the likewise well-known sets of 3L-CVRP benchmark instances by Gendreau et al. (2006) and by Tarantilis et al. (2009).

To provide a sufficiently large set of 3L-VRPCB benchmark instances each of the mentioned 95 VRPCB instances is extended to a 3L-VRPCB instance by the following procedure:

- Each of the 95 VRPCB instances is considered as the base of a new 3L-VRPCB instance, i.e. the data of a VRPCB instance are completely reused in the corresponding 3L-VRPCB instance. The scalar demand of a linehaul or backhaul customer is interpreted as the weight d_i of the corresponding box set. The vehicles' capacity is now seen as the weight limit D (see constraint C1).
- We consider the 27 3L-CVRP instances introduced by Gendreau et al. (2006) (with up to 100 customers) and the 12 3L-CVRP instances proposed by Tarantilis et al. (2009). Each of the 95 VRPCB instances is merged with exactly one of the 39 3L-CVRP instances. Let a VRPCB instance have l linehaul and b backhaul customers. Then a 3L-CVRP instance for merging is determined such

that its number of customers is not smaller than the maximum of l and b.

- Given a VRPCB instance and its associated 3L-CVRP instance, each of the linehaul customers of the new 3L-VRPCB instance receives the box set including the fragility attributes of a customer of the 3L-CVRP instance. The same procedure is applied to the backhaul customers. The dimensions of the loading spaces of 3L-CVRP instances and support parameter a (see constraint C4) are adopted (i.e., a is set to 0.75).
- The maximal number of admitted vehicles v_{max} per instance is determined as the number of routes of the solution that is determined by the constructive heuristic which was proposed in Subsection 4.2. In this way, a feasible solution with no more than v_{max} routes can be relatively easily obtained.
- The set of 3L-VRPCB instances derived from VRPCB instance set GJB is denoted by 3L-GJB whereas the set of instances derived from VRPCB instance set TV is denoted by 3L-TV.

The new 3L-VRPCB instances are characterized by some important data that are summarized in Table 3. The mean number of boxes per vehicle stands for the mean quotient (volume of loading space) / (mean box volume). The instances will be provided at the website http://www.mansci.ovgu.de.

Table 3: Characteristics of the 95 new 3L-VRPCB benchmark instances.

3L-VRPCB instance set	VRPCB base set	No. of customers			Percentage of backhaul customers (%)			Total no. of boxes			No. of boxes per vehicle
		min	max	mean	min	max	mean	min	max	mean	mean
3L-GJB	GJB	25	150	81.5	20	50	33.7	46	461	170.7	16.7
3L-TV	TV	21	100	59.5	17.2	50	33.1	37	309	122.5	18.1
All	All	21	150	73.9	17.2	50	33.5	37	461	153.9	17.6

5.2 Parameter setting

The parameter setting for the experiments is specified in Table 4 to 6. The same parameterization of the routing procedures is used for all variants of the hybrid algorithms (some alternative settings are tried in Section 5.5). Further details on the ALNS parameters can be found in Ropke and Pisinger (2006a, p. 13 f). All parameter values were determined based on limited computational experiments using a trial and error strategy.

Table 4: Parameter setting for the ALNS routing procedure.

Parameter	Description	Value
r_{min}	lower bound of no. of removed customers	$0.04 \cdot (l+b)$
r_{max}	upper bound of no. of removed customers	$0.4 \cdot (l+b)$
w	start temperature control parameter	0.005
c	rate of geometrical cooling	0.9999
σ_1, σ_2, σ_3	score increments	33, 9, 13
r	reaction factor	0.1
σ_{max}	length of segment list	100
$p(Rh_R)$, $p(Rh_S)$	initial probability of Random / Shaw removal	0.3, 0.4
$p(Rh_W)$, $p(Rh_T)$	initial probability of Worst / Tour removal	0.1, 0.2
$p(Ih_G)$, $p(Ih_{R2})$, $p(Ih_{R3})$	initial probability of Greedy / Regret-2 / Regret-3 insert	0.1, 0.6, 0.3
w_{r1}, w_{r2}	weights of relatedness formula for Shaw removal	9, 2

Table 5: Parameter setting for the VNS routing procedure.

Parameter	Description	Value
α	initial value for the penalty coefficient in the objective function	500
β	final value for the penalty coefficient in the objective function	1000
ψ	step size for increasing the penalty coefficient	0.05

The parameters of the packing procedure for the variants 3D-LNS-H/L and 3D-VNS-H/L are shown in Table 6 (see also Bortfeldt, 2012).

Table 6: Parameter setting for packing procedure (nc: no. of customers).

Parameter	Description	Values for 3D-LNS-H	Values for 3D-LNS-L	Values for 3D-VNS-H	Values for 3D-VNS-L
maxApCalls	Max. no. of calls of procedure add_placement	nc ≤ 50: 3000 51 ≤ nc ≤ 99: 1500 100 ≤ nc: 750	nc ≤ 50: 120 51 ≤ nc ≤ 99: 60 100 ≤ nc: 30	3000	1500
maxBoxRankDiff	Max. tolerated rank difference of boxes	2	2	2	2
maxRefPoints	Max. number of admitted reference points	3	3	3	3

The maximum number of calls of the recursive procedure add_placement determines to a large extent whether the tree search is carried out thoroughly or not. For the ALNS algorithm, the values of the parameter *maxApCalls* depend on the instance size, i.e. the number of customers; however, the high effort is uniformly chosen 25 times greater than the low effort for a packing check. For the VNS algorithm the values of parameter *maxApCalls* are fixed for both effort variants. Preliminary tests taught us that only relatively high values of parameter *maxApCalls* yield acceptable results of the VNS algorithm; therefore the high effort is chosen only two times larger than the low effort of a packing check. The remaining packing parameters are kept constant for all 3D algorithm variants.

The maximum computing time per instance and single run is set to 3 minutes for the two variants 1D-ALNS and 1D-VNS. For the four variants 3D-ALNS-H/L and 3D-VNS-H/L, this time limit depends on the instance size. The limit is set to 10 minutes for instances with maximal 50 customers, while it is set to 20 minutes for instances with 51 to 99 customers. Finally, the time limit is chosen as 30 minutes for instances with at least 100 customers. For the 3D variants of the hybrid algorithms, the best solution is recorded after 3 minutes (reduced calculation time) and after the time limit has been reached (full calculation time). For the 1D-LNS and 1D-VNS variants, five independent runs are performed whereas only three independent runs are conducted per instance for the 3L-VRPCB variants of the algorithms due to the high computational burden. In the 3L-CVRP test using the ALNS hybrid algorithm 20 runs are performed per instance. The time limit per run is specified in the same manner as for the 3L-VRPCB, e.g. depending on number of customers (nc) we use 10 (nc ≤ 50), 20 (51 ≤ nc ≤ 99) or 30 (100 ≤ nc) minutes as time limit.

5.3 Computational results for the one-dimensional VRPCB

The overall test results for the algorithm variants 1D-ALNS and 1D-VNS and the instance sets GJB and TV are shown in Table 7. The name of the instance set is listed in the leftmost column. The best total travel distances and their gaps to the best known solutions are shown for each of the algorithm variants 1D-ALNS and 1D-VNS. They are also indicated for 1D-ALNS with a time limit of only 30 seconds per run and for the VRPCB-ALNS algorithm by Ropke and Pisinger (2006b). For a single instance, the gap is determined as (*ttd-best* − *ttd-best-known*) / *ttd-best-known* (in %) where *ttd-best* is the best reached travel distance (over the five runs) and *ttd-best-known* is the best known travel distance for the given instance. Best known travel distances are taken from Vidal et al. (2014), Zachariadis and Kiranoudis (2012) and Arráiz and Palhazi Cuervo (2011). The best total travel distances and gaps are averaged over all instances of the respective instance set; the last line shows average values over both sets.

As the results indicate, the ALNS algorithm reaches nearly the solution quality of the best available methods for the (1D) VRPCB even for the smaller time limit. In particular, a high solution quality was achieved although the ALNS has been rather simplified compared to the original procedures by Ropke and Pisinger (2006a/b). The VNS algorithm performs worse and misses the best known values by approximately 3.18% on average over the 95 instances of the GJB and TV instance sets.

Table 7: Overall results for 1D test.

Instance set	1D-ALNS (3 min)		1D-ALNS (30 s)		Ropke/Pisinger (2006b)		1D-VNS (3 min)	
	ttd-best	gap (%)	ttd-best	gap (%)	ttd-best	gap (%)	ttd-best	gap (%)
GJB	290889.94	0.09	291088.01	0.16	290896.73	0.09	300386.90	3.14
TV	701.48	0.10	701.60	0.12	701.00	0.04	724.52	3.26
All	190087.63	0.10	190216.95	0.15	190091.90	0.07	196293.65	3.18

5.4 Computational results for the 3L-VRPCB

The detailed results for the instance sets 3L-GJB and 3L-TV are presented in the Tables 8 and 9. These values were calculated by means of the four algorithm variants 3D-ALNS-H/L and 3D-VNS-H/L within the specified time limits. In the leftmost column of each table, the name of the 3L-VRPCB instance is listed. The best and the average total travel distance over the three runs are listed for each of the four algorithm variants in the following eight columns. The best and the mean total travel distances are averaged over all instances of the respective set in the last row.

Further average results for the four 3D variants of the hybrid algorithms are indicated in Table 10. For each algorithm variant and for full and reduced calculation time two gaps are calculated. The first gap (*gap-best*) is determined per instance as (*ttd-best* − *ttd-best-all*) / *ttd-best-all* (in %) where *ttd-best* is the best reached travel distance over the three runs of the algorithm and *ttd-best-all* is the best calculated travel distance for the given instance calculated by all four algorithms studied here (within full calculation time). The second gap (*gap-avg*) is determined per instance as (*ttd-avg* − *ttd-best-all*) / *ttd-best-all* (in %) where *ttd-best-all* is defined as before and *ttd-avg* is the average total travel distance over the three runs of the algorithm. Both gaps are then averaged over the instance sets 3L-GJB and 3L-TV as well as over the set of all 95 instances.

Moreover, the fraction (*3D-ttd*/*1D-ttd*) indicates the increase of the travel distance for the 3D backhaul problem instances compared with the corresponding 1D problem instances. For each 3D algorithm variant the distance *3D-ttd* is calculated as the average travel distance over all instances of the respective set as well as all runs within full calculation time. For the corresponding 1D algorithm variant the distance *1D-ttd* is calculated similarly.

Summarizing the results we can state that the travel distances increase significantly if the 3L-VRPCB instances are solved instead of the corresponding 1D VRPCB instances. The best results were achieved with the algorithm variant 3D-ALNS-H using full calculation time. In this situation, the travel distances increase by 33% on average over the 95 test instances compared to 1D-ALNS. The mean increase of the travel distances is even higher for other the 3D algorithm variants.

The effect of time limits depends strongly on the 3D algorithm variant. Significant improvements for full compared to reduced (3 minutes) calculation time result for the variant 3D-ALNS-H. Considering the *gap-avg* values, a reduction of 6.1%-points of the travel distance is achieved for the 3L-GJB instance set, while the saving for the 3L-TV instance set is 3.7%-points. The improvements by longer computing times are much smaller or even no reductions can be observed at all for the other three algorithm variants.

The ALNS variants outperform the VNS variants. The 3D-ALNS-H variant achieves an improvement of 7.1%-points for the 3L-GJB set and of 4.6%-points travel distance for the 3L-TV set compared to 3D-VNS-H if best solutions are considered; the improvement for the average values is 8.7%-points for the 3L-GJB set and 7.2%-points for the 3L-TV set. The improvements are even higher if the algorithm variants with low specific packing effort are compared. The ALNS variants are characterized by neighborhood structures that change larger portions of a given solution compared to the VNS variants. Therefore, an additional experiment with 3D-ALNS-H was carried out in which the parameters r_{min} and r_{max} were set to $0.01 \cdot (l+b)$ and $0.05 \cdot (l+b)$, respectively (cf. Table 4). The results for this ALNS with smaller changes from a solution to its neighbors are indicated in the last lines of Table 10. They are in line with the interpretation that applying neighborhood structures leading to more drastical solution changes is favorable. However, this phenomenon remains a subject of further research.

Finally, we look at the influence of the specific packing effort. For the ALNS variants, the effect of the specific packing effort depends on computing times. For reduced calculation time of 3 minutes the variants with high and low specific packing effort, respectively, are nearly on a par. However, for full calculation time the variant with high specific packing effort achieves significant improvements of the travel distances of 2.8%-points regarding best values and 2.9%-points regarding average values if all 95 instances are included. Hence, more thorough packing checks pay off if enough computing time is available. The VNS variant with low specific packing effort performs rather poor.

Table 8: Results for instance set 3L-GJB.

Instance	3D-ALNS-H		3D-ALNS-L		3D-VNS-H		3D-VNS-L	
	ttd-best	ttd-avg	ttd-best	ttd-avg	ttd-best	ttd-avg	ttd-best	ttd-avg
3L-A1	230547.48	230547.48	230547.48	230547.50	230547.51	234945.55	230547.51	234945.55
3L-A2	185878.89	185878.89	190744.67	190744.67	195031.47	196798.75	195031.47	196798.75
3L-A3	185797.69	185797.69	198113.41	198385.55	189601.18	190085.28	189601.18	191624.39
3L-A4	201004.16	201764.17	203284.16	203753.67	204947.21	204947.21	204947.21	204947.21
3L-B1	239080.14	239080.14	250712.78	250712.80	239080.16	239080.16	239080.16	239080.16
3L-B2	214911.06	219068.45	222308.25	226654.13	232464.93	234979.36	235329.86	244240.24
3L-B3	209133.97	209331.63	208017.83	210181.69	239701.36	242863.95	230816.12	237652.94
3L-C1	262798.06	262798.06	272184.59	272184.59	266577.78	267706.10	269547.33	271472.84
3L-C2	230569.17	230569.19	233659.95	233659.95	235343.09	239806.19	235343.09	245061.73
3L-C3	218968.13	220025.45	221560.66	221560.67	228125.45	234426.22	230616.28	240322.12
3L-C4	226661.41	226661.42	236006.66	236006.67	238128.33	242655.01	240726.30	241245.40
3L-D1	329091.88	329647.28	330749.13	331383.28	357967.16	357967.16	357967.16	357967.16
3L-D2	318301.25	318301.25	318301.25	318301.25	341092.90	341092.90	341092.90	341092.90
3L-D3	257985.17	257985.19	269736.72	269736.72	259946.79	275579.12	259946.79	275579.12
3L-D4	251251.02	251421.63	251251.00	252973.77	260367.67	267536.44	261128.69	267790.12
3L-E1	250467.34	250467.33	258501.02	258501.02	278761.92	288567.07	280195.89	289045.06
3L-E2	246826.84	247729.92	251800.42	251922.77	260754.62	260754.62	260754.62	261220.39
3L-E3	265770.06	266807.38	275723.41	277617.91	291810.20	303328.85	296244.19	302824.81
3L-F1	299275.47	299904.78	312122.03	312793.59	352054.21	353515.99	438014.57	438014.57
3L-F2	284334.09	285204.72	291699.03	291699.03	310423.00	323494.97	312364.73	313501.38
3L-F3	311427.91	311427.91	329945.97	329945.97	326548.97	337201.08	365542.60	369959.05
3L-F4	300348.00	300709.69	314501.44	314911.31	318957.83	326360.22	329087.27	341951.86
3L-G1	338442.25	338914.22	344575.94	345210.50	348042.17	351896.43	348042.17	352261.56
3L-G2	261518.30	265472.56	263726.13	263858.16	281299.10	292095.55	274026.33	296348.17
3L-G3	400059.50	400414.78	400059.53	400407.50	423602.82	428043.06	423602.82	428043.06
3L-G4	322648.88	324103.16	330143.72	330756.84	348848.02	353064.37	354304.00	360172.21
3L-G5	268092.16	276378.59	281857.50	288113.50	274443.04	285868.75	406354.83	406354.83
3L-G6	409300.16	409874.09	409300.16	409300.16	430132.17	442154.70	430132.17	442154.70
3L-H1	341086.84	343181.88	343734.50	344531.97	353113.43	363008.68	353113.43	362038.14
3L-H2	278621.38	283724.72	287728.97	288044.03	294819.81	303416.48	410644.02	410644.02
3L-H3	415112.25	415587.41	414489.00	414489.00	432335.09	437226.47	432335.09	437226.47
3L-H4	351044.44	355565.25	352675.03	353244.47	381516.62	392568.38	365695.64	374613.91
3L-H5	282750.63	287337.97	287583.06	288656.25	290006.37	294198.32	293714.98	304249.84
3L-H6	407461.97	409330.84	407139.28	409040.97	434887.66	442260.51	434887.66	442260.51
3L-I1	391147.13	391794.28	402370.13	406587.00	423009.30	430422.67	418716.46	426051.94
3L-I2	318286.75	320225.78	333276.88	334886.28	320703.78	344208.99	351396.26	353749.79
3L-I3	464401.28	466438.75	464401.28	466127.72	487336.48	507992.42	487336.48	507992.42
3L-I4	374164.56	376894.38	388487.00	389910.97	400891.43	428678.46	673996.83	673996.83
3L-I5	321289.44	330322.09	339548.63	344879.63	378234.81	386392.52	545356.17	545356.17
3L-J1	589913.56	592596.50	589842.25	591695.00	640153.66	647862.22	640153.66	647862.22
3L-J2	500704.84	502688.91	504479.19	506932.66	519829.34	529083.15	756930.77	756930.77
3L-J3	458536.56	466793.75	468467.19	470254.16	544887.40	556563.49	685199.56	685199.56
3L-J4	496257.56	500863.72	502313.41	503195.97	561597.65	573724.69	718076.87	718076.87
3L-K1	534588.75	538198.88	562439.50	565405.44	550875.34	561845.94	550442.03	569567.76
3L-K2	500566.94	503389.00	535729.13	544665.75	549489.81	563931.03	839897.60	839897.60
3L-K3	432218.28	436771.50	469009.88	474073.47	442364.31	452055.52	441086.55	458710.86
3L-K4	622303.50	623821.81	625863.00	628139.75	675005.86	684170.14	675005.86	684170.14
3L-L1	627982.31	636345.69	632639.38	650673.00	774441.65	818475.73	1087916.81	1087916.81
3L-L2	625359.50	630132.06	692911.06	698211.94	722570.17	727661.71	1037759.68	1037759.68
3L-L3	616702.19	620349.94	624222.13	628005.81	712831.94	727138.10	1076429.02	1076429.02
3L-L4	601636.63	609441.06	637134.00	643009.31	621892.06	630324.43	1051550.75	1051550.75
3L-L5	589264.56	594466.19	607845.06	614668.00	708002.88	715963.73	953463.71	953463.71
3L-M1	609447.81	610786.88	628310.19	637246.94	624886.81	661596.09	852149.08	852149.08
3L-M2	650136.00	665514.00	673173.13	681412.81	696740.59	705173.82	971046.74	971046.74
3L-M3	654280.31	671713.50	706788.44	719305.75	707342.27	711596.53	682537.82	700250.35
3L-M4	526784.31	532249.75	581038.50	610062.06	514398.31	522610.04	717661.74	717661.74
3L-N1	544695.69	568288.25	602260.31	624841.19	638276.24	641341.09	806952.93	806952.93
3L-N2	816836.75	819321.06	817204.19	820746.50	868739.13	878581.64	868739.13	878581.64
3L-N3	679652.31	686907.50	710819.88	717101.00	827803.93	849421.33	831912.56	843106.63
3L-N4	632841.50	634401.56	719410.44	751479.50	556744.71	575218.96	802576.27	802576.27
3L-N5	879287.63	884558.31	877732.88	881446.00	960019.11	970410.88	960019.11	970410.88
3L-N6	675610.13	680373.56	694930.31	699483.31	721347.46	740425.69	1144492.57	1144492.57
Average	**408572.01**	**411881.67**	**422373.10**	**426198.04**	**440349.94**	**449876.85**	**518702.94**	**523945.43**

Table 9: Results for instance set 3L-TV.

Instance	3D-ALNS-H		3D-ALNS-L		3D-VNS-H		3D-VNS-L	
	ttd-best	ttd-avg	ttd-best	ttd-avg	ttd-best	ttd-avg	ttd-best	ttd-avg
3L-EIL_22_50	373.92	373.92	389.84	389.84	403.89	403.89	403.89	403.89
3L-EIL_22_66	389.06	389.66	398.82	400.02	398.28	406.67	398.28	406.67
3L-EIL_22_80	396.22	396.22	396.22	396.22	403.95	412.31	403.95	408.80
3L-EIL_23_50	731.96	740.59	731.96	731.96	743.82	751.87	737.07	753.65
3L-EIL_23_66	716.99	716.99	721.46	721.46	737.49	758.41	737.49	758.41
3L-EIL_23_80	700.72	700.72	719.94	728.61	711.24	741.82	711.24	739.64
3L-EIL_30_50	606.12	606.12	610.84	610.84	638.60	640.18	634.87	689.35
3L-EIL_30_66	621.30	621.30	625.23	625.23	663.69	666.89	629.58	655.52
3L-EIL_30_80	758.66	758.82	760.41	764.73	845.02	858.49	845.02	858.49
3L-EIL_33_50	892.24	892.24	892.24	892.24	892.24	952.94	892.24	957.17
3L-EIL_33_66	1122.67	1126.91	1130.88	1144.85	1220.50	1252.30	1246.02	1270.06
3L-EIL_33_80	1160.35	1170.21	1178.36	1178.36	1248.31	1265.28	1249.41	1252.32
3L-EIL_51_50	709.28	711.58	722.02	725.52	733.93	758.05	727.73	758.30
3L-EIL_51_66	647.31	649.69	650.96	654.23	683.33	697.19	666.19	684.49
3L-EIL_51_80	735.47	742.45	741.12	741.82	775.74	803.86	985.21	985.21
3L-EIL_A76_50	873.32	878.90	908.23	915.52	911.25	949.44	913.61	932.18
3L-EIL_A76_66	954.56	957.29	986.14	987.78	981.24	1016.38	1015.41	1068.59
3L-EIL_A76_80	1057.78	1071.93	1066.68	1077.03	1109.31	1135.00	1134.40	1152.66
3L-EIL_B76_50	897.11	898.32	920.54	925.42	1050.65	1061.41	1036.81	1068.39
3L-EIL_B76_66	974.04	975.06	998.06	1005.42	1037.06	1057.46	1055.31	1084.19
3L-EIL_B76_80	1072.92	1082.23	1078.25	1085.02	1104.46	1143.42	1100.76	1126.44
3L-EIL_C76_50	927.54	928.78	943.07	943.07	989.88	999.64	1002.50	1041.53
3L-EIL_C76_66	964.49	973.66	983.39	986.21	981.24	1016.38	1072.10	1088.97
3L-EIL_C76_80	1064.81	1068.70	1061.65	1069.04	1073.25	1114.20	1476.99	1476.99
3L-EIL_D76_50	927.54	928.78	943.07	943.07	989.88	999.64	1002.50	1041.53
3L-EIL_D76_66	964.49	973.66	983.39	986.21	981.24	1016.38	1072.10	1088.97
3L-EIL_D76_80	1064.81	1068.70	1061.65	1069.04	1073.25	1114.20	1476.99	1476.99
3L-EIL_A101_50	946.70	950.48	993.45	999.06	961.64	1067.22	1361.32	1361.32
3L-EIL_A101_66	1231.89	1236.05	1259.41	1270.49	1266.06	1362.85	1421.94	1461.74
3L-EIL_A101_80	1398.38	1404.73	1442.51	1449.00	1464.17	1471.93	1475.06	1515.48
3L-EIL_B101_50	1421.69	1422.98	1421.67	1425.18	1497.03	1514.77	1497.03	1514.77
3L-EIL_B101_66	1225.19	1229.58	1233.47	1238.90	1303.35	1364.70	1309.64	1365.05
3L-EIL_B101_80	1155.50	1189.12	1400.20	1430.07	1189.42	1225.02	1515.02	1515.02
Average	899.55	904.13	919.85	924.59	941.35	969.70	1006.29	1029.18

Table 10: Further overall results for the 3L-GJB and 3L-TV instance sets.

Instance set	Algorithm: 3D-ALNS-H				
	full time			reduced time	
	3D-ttd / 1D-ttd	gap-best %	gap-avg %	gap-best %	gap-avg %
3L-GJB	1.36	0.27	1.00	5.39	7.11
3L-TV	1.27	0.02	0.47	2.42	4.12
All	1.33	0.19	0.82	4.36	6.07
Instance set	**Algorithm: 3D-ALNS-L**				
	full time			reduced time	
	3D-ttd / 1D-ttd	gap-best %	gap-avg %	gap-best %	gap-avg %
3L-GJB	1.41	3.48	4.26	5.44	6.54
3L-TV	1.30	2.13	2.60	2.88	3.54
All	1.37	3.01	3.68	4.55	5.50
Instance set	**Algorithm: 3D-VNS-H**				
	full time			reduced time	
	3D-ttd / 1D-ttd	gap-best %	gap-avg %	gap-best %	gap-avg %
3L-GJB	1.46	7.38	9.72	10.02	12.74
3L-TV	1.33	4.67	7.67	5.09	8.16
All	1.42	6.44	9.01	8.30	11.15
Instance set	**Algorithm: 3D-VNS-L**				
	full time			reduced time	
	3D-ttd / 1D-ttd	gap-best %	gap-avg %	gap-best %	gap-avg %
3L-GJB	1.69	22.89	24.40	23.80	25.32
3L-TV	1.42	10.98	13.60	11.24	13.78
All	1.59	18.75	20.65	19.44	21.31
Instance set	**Algorithm: 3D-ALNS-H**		**(r_{min} = 0.01 (l+b), r_{max} = 0.05 (l+b))**		
	full time			reduced time	
	3D-ttd / 1D-ttd	gap-best %	gap-avg %	gap-best %	gap-avg %
3L-GJB	1.40	3.01	4.91	10.18	12.43
3L-TV	1.31	2.85	5.03	7.59	9.84
All	1.37	2.96	4.95	9.28	11.53

5.5 Computational experiments with individual ALNS components

In the following, the impact of different components of the hybrid ALNS algorithm on the solution quality is studied. For that purpose nine experiments are performed with the 3D-ALNS-H variant.

In each of the first six experiments the routing procedure is changed regarding one of its components. In these experiments we want to research whether the results are significantly deteriorated or improved if routing components are modified, removed or added. In particular, this question is dealt with regarding important operators (Shaw removal, Regret-4 insertion) and the learning layer. In each of the last three experiments one component of the routing/packing integration is switched off or modified. Here we want to check the efficiency of the chosen coupling mechanism between routing and loading.

All nine experiments are executed using the 95 3L-VRPCB instances introduced above. Each instance is run three times with the time limit specified in Section 5.2. For each experiment the gaps *gap-best* and *gap-avg* are averaged over all 95 3D instances of the sets 3L-GJB and 3L-TV. These gaps are calculated as defined in Section 5.4 and by means of the *ttd-best-all* values as determined in Section 5.4 (even if better travel distances result for some instances by some of the algorithm variants tested here). Table 11 describes some details of the experiments and their outcome. For convenience, the gap results for the original 3L-ALNS-H algorithm are shown again (see line 0).

Table 11: Experiments with individual components of 3L-ALNS-H.

#	Modification of heuristic 3D-ALNS-H	Results for 3L-GJB *and* 3L-TV	
		gap-best (%)	gap-avg (%)
0	Original version of 3L-ALNS-H (as in Table 10).	0.19	0.82
1	Pure LNS without learning layer, i.e. fixed selection probabilities for all removal and insertion heuristics.	0.13	0.73
2	Shaw removal modified: relatedness is determined only by factor location, i.e. by distance between two customers.	0.17	0.88
3	Shaw removal modified: relatedness is determined by three factors, namely location, volume and weight and calculated as weighted sum of distance, weight and volume difference, respectively, between two customers.	0.27	0.81
4	Tour removal eliminated.	0.38	1.19
5	Regret-4 insertion added.	0.30	1.06
6	Noise applied to objective function (see Ropke and Pisinger 2006a, p. 10 f).	0.27	1.04
7	Cache for customer sequences switched off (see Section 4.3).	1.76	3.21
8	Principle "Evaluating first, packing second" switched off, i.e. insertions are not sorted by cost before 3D packing checks are made (see Section 4.1.2, Fig. 5).	1.65	2.75
9	Weight test carried out *only after* 3D packing check (see Section 4.1.2, Fig. 5).	0.27	0.94

The comparison of the gaps in Table 11 shows very clearly that the modifications 1 to 6 of the routing procedure (regarding learning layer, operators and noise) have only a negligible impact on the solution quality. This general result seems to justify our approach to implement the 3L-ALNS algorithm in a rather straightforward fashion (in particular, a pure LNS algorithm would suffice). While modification 9 has also hardly any impact the modifications 7 and 8 lead to a significant loss of solution quality. Hence, the core components for coupling routing and packing (customer sequence cache, principle "Evaluation first, packing second") turn out to be key factors of performance.

5.6 Computational results for 3L-CVRP

Since the 3L-CVRP is a special case of the 3L-VRPCB, having only linehaul customers, the ALNS hybrid algorithm, i.e. variant 3L-ALNS-H, is applied now to 3L-CVRP benchmark instances. All the parameters (in particular time limits) have been specified in Section 5.2. The results for the 27 instances by Gendreau et al. (2006) and the 12 instances by Tarantilis et al. (2009) are presented in Tables 12 and 13. Both tables are built similarly. In the leftmost column the instances (including no. of customers) are specified. In the following columns best and (where necessary) average travel distances over the performed runs are indicated for some high grade 3L-CVRP solution methods (cf. Section 2.4) are indicated. Best values are set in bold. Mean travel distances are shown in the last lines.

Considering the 27 instances by Gendreau et al. (2006) 3D-ALNS-H reaches good, but not outstanding results. For 8 of these instances solutions with best known travel distances were found. However, 3D-ALNS-H performs very well for the particularly challenging 12 instances by Tarantilis et al. (2009). New best solutions were generated for 4 instances and 3D-ALNS-H achieved better re-

sults in terms of best and average solution quality than all compared methods. The run times of 3D-ALNS-H remain moderate. For the 27 instances by Gendreau et al., e.g., our algorithm needs 844 seconds CPU time on average while Wang et al. (2010) publish a mean running time of 820 seconds and Wisniewski et al. (2011) indicate a run time of 930 seconds.

Table 12: Results for 27 3L-CVRP instances by Gendreau et al. (2006).

Instance (nc)	Wang et al. 2010		Wisniewski et al. 2011		Bortfeldt 2012		Zhu et al. 2012		Ruan et al. 2013		3D-ALNS-H	
	ttd-best	ttd-avg	ttd-best	ttd-avg	ttd-best	ttd-avg	ttd-best	ttd-avg	ttd-best	ttd-avg	ttd-best	ttd-avg
1 (15)	**301.74**	301.77	304.13	304.84	302.02	302.02	302.02	302.23	303.21	-	302.02	302.02
2 (15)	**334.96**	334.96	**334.96**	334.96	**334.96**	334.96	**334.96**	334.96	**334.96**	-	**334.96**	334.96
3 (20)	387.34	387.91	**381.37**	384.90	392.63	401.44	394.00	409.44	398.05	-	388.09	392.63
4 (20)	**437.19**	438.59	**437.19**	437.19	**437.19**	437.54	**437.19**	439.98	440.68	-	**437.19**	438.70
5 (21)	**436.48**	440.23	442.21	449.66	443.61	451.03	443.61	447.36	452.56	-	443.61	445.40
6 (21)	498.32	499.48	499.02	501.77	**498.16**	498.38	498.32	499.99	498.56	-	**498.16**	500.30
7 (22)	**767.46**	771.09	769.68	770.22	769.68	772.49	768.85	773.31	790.23	-	769.68	769.68
8 (22)	**803.98**	805.95	806.25	808.14	810.89	821.35	805.35	807.59	820.67	-	808.55	808.67
9 (25)	**630.13**	630.90	**630.13**	630.13	**630.13**	645.81	**630.13**	630.13	635.50	-	**630.13**	630.13
10 (29)	826.39	832.46	822.46	825.86	**820.35**	827.29	834.80	839.75	836.21	-	**820.35**	821.78
11 (29)	**768.25**	781.85	781.17	788.77	803.61	815.62	786.19	790.47	825.75	-	783.97	787.22
12 (30)	**610.23**	614.78	**610.23**	610.43	614.59	630.46	612.25	615.05	626.59	-	**610.23**	614.62
13 (32)	2697.70	2715.82	2693.24	2713.05	**2645.95**	2694.81	2719.65	2732.85	2739.80	-	2681.02	2715.19
14 (32)	1428.99	1456.13	1390.18	1425.18	**1368.42**	1413.59	1403.75	1460.34	1469.38	-	1409.22	1450.95
15 (32)	1352.94	1371.26	1350.85	1370.41	**1341.14**	1355.50	1371.58	1386.75	1369.69	-	1349.68	1376.71
16 (35)	**698.61**	699.54	**698.61**	699.49	**698.61**	705.05	**698.61**	698.69	703.15	-	**698.61**	701.45
17 (40)	871.63	875.19	871.24	871.71	**866.40**	917.96	**866.40**	869.96	872.05	-	**866.40**	871.96
18 (44)	1227.07	1248.28	1227.08	1248.09	**1207.72**	1228.98	1231.80	1252.67	1250.86	-	1212.59	1260.35
19 (50)	762.47	776.35	755.56	763.44	**741.74**	753.87	767.19	777.96	780.37	-	747.48	767.48
20 (71)	583.45	593.17	**579.18**	586.00	587.95	596.42	597.75	600.82	605.59	-	590.98	610.36
21 (75)	1094.78	1121.60	**1085.53**	1099.24	1090.22	1107.00	1133.29	1140.11	1119.45	-	1107.83	1147.69
22 (75)	1170.89	1176.76	1151.48	1159.85	**1147.80**	1171.49	1193.82	1199.14	1167.28	-	1175.36	1211.77
23 (75)	1137.90	1148.02	**1119.08**	1128.71	1130.54	1135.46	1163.10	1176.07	1171.77	-	1145.14	1163.20
24 (75)	1132.05	1144.56	1116.57	1126.55	**1116.13**	1128.82	1151.13	1161.87	1136.27	-	1138.22	1164.07
25 (100)	1434.00	1457.09	**1389.95**	1405.00	1407.36	1428.80	1414.85	1442.62	1426.34	-	1419.64	1482.73
26 (100)	1606.85	1616.61	1594.45	1612.31	1600.35	1625.31	1607.85	1614.56	**1585.46**	-	1586.06	1618.86
27 (100)	1551.68	1574.23	1537.10	1560.06	**1529.86**	1550.85	1553.77	1571.38	1562.18	-	1576.86	1599.82
Average	946.43	956.10	939.96	948.74	938.44	953.79	952.67	962.08	960.10	-	945.63	962.54

Table 13: Results for 12 3L-CVRP instances by Tarantilis et al. (2009).

Instance (nc)	Tarantilis et al. 2009		Bortfeldt 2012		Zhu et al. 2012		3D-ALNS-H	
	ttd-best	ttd-avg	ttd-best	ttd-avg	ttd-best	ttd-avg	ttd-best	ttd-avg
50_1 (50)	1457.78	-	1438.60	1456.30	**1433.08**	1451.14	1438.53	1458.38
50_2 (50)	2257.60	-	**2192.73**	2225.20	2201.30	2222.50	2203.99	2224.81
50_3 (50)	1838.40	-	**1744.26**	1821.90	1799.78	1817.52	1775.00	1812.00
75_1 (75)	2059.32	-	**2039.48**	2067.47	2044.53	2078.00	2045.71	2064.93
75_2 (75)	3279.16	-	3023.22	3078.92	3099.78	3166.69	**3023.07**	3058.26
75_3 (75)	2508.17	-	**2420.20**	2534.36	2479.09	2540.58	2421.80	2517.91
100_1 (100)	2690.23	-	2595.22	2645.02	**2588.31**	2613.04	2681.02	2714.99
100_2 (100)	4342.64	-	4239.55	4269.22	4205.44	4237.80	**4191.75**	4245.25
100_3 (100)	4190.92	-	4157.38	5007.87	4078.21	4263.59	**3725.61**	3856.49
125_1 (125)	3298.22	-	3240.76	3294.00	**3240.40**	3260.23	3301.91	3356.02
125_2 (125)	5788.12	-	**5370.64**	5563.91	5379.70	5450.58	5469.54	5590.94
125_3 (125)	5177.97	-	5222.21	6401.59	4800.27	5078.29	**4598.76**	5121.61
Average	3240.71	-	3140.35	3363.81	3112.49	3181.66	3073.06	3168.47

6 Conclusions and future research

In this paper, the vehicle routing problem with clustered backhauls has been extended to an integrated vehicle routing and loading problem with 3D rectangular items to be transported and homogeneous vehicles with a rectangular 3D loading space. Additionally, several loading constraints are included in the problem formulation that occur frequently in freight transportation. Two hybrid algorithms for solving the 3L-VRPCB have been proposed. In the first algorithm, an ALNS procedure serves for routing vehicles while a VNS procedure is responsible for this task in the second algorithm. Both of the algorithms include the same tree search procedure for packing items in a loading space that was adopted from a previous hybrid algorithm for 3L-CVRP. Other key features concern the integration of routing and packing, namely the "evaluating first, packing second" technique and the use of a cache

for routes, and serve to ensure a reasonable computational effort. The hybrid algorithms were tested by means of 95 newly introduced 3L-VRPCB instances that were derived from well-known VRPCB benchmark instances. In the calculations, plausible results were reached for both the algorithms. In particular, an increase of travel distance of 30% or more has been observed if the 3D backhaul problem is solved instead of the corresponding 1D backhaul problem. Of course, other findings may result by other load settings regarding mean volume, shape of boxes etc. and this could be a subject of further investigations. However, it seems that modeling and calculating real-world scenarios with backhauls as three-dimensional backhaul problems will often lead to more realistic results. High-quality results were also achieved by means of the first hybrid algorithm for two well-known sets of benchmark instances of the 3L-CVRP being a special case of the 3L-VRPCB.

There are several directions for future research. First, we want extend further routing problems to integrated routing and loading problems as, e.g., the pickup and delivery problem. Including time windows into the problem setting of the present paper is also an interesting avenue for future research. In addition, we believe that it is also worth to study the cooperative vehicle routing problem in Sprenger and Mönch (2012) with additional backhauls and loading constraints. This problem is strongly influenced by a real-world setting where backhauls occur.

References

Arráiz, E, Palhazi Cuervo, AD (2011): An Iterated Local Search Algorithm for the Vehicle Routing Problem with Backhauls. *Proceedings of the 9th Metaheuristic International Conference (MIC)*, 339-343.

Bortfeldt, A (2012): A Hybrid Algorithm for the Capacitated Vehicle Routing Problem with Three-Dimensional Loading Constraints. *Computers & Operations Research*, 39:2248-2257.

Bortfeldt, A, Homberger, J (2013): Packing First, Routing Second - a Heuristic for the Vehicle Routing and Loading Problem. *Computers & Operations Research*, 40:873-885.

Bortfeldt, A, Hahn, T, Mönch, L (2013): A Hybrid Algorithm for the Vehicle Routing Problem with Backhauls and 3D Loading Constraints (Extended Abstract). *Proceedings of the 10th Metaheuristic International Conference (MIC),* 59-61.

Brandao, J (2006): A New Tabu Search Algorithm for the Vehicle Routing Problems with Backhauls. *European Journal of Operational Research*, 173:540-555.

Clarke, G, Wright, JW (1964): Scheduling of Vehicles from a Central Depot to a Number of Delivery Points, *Operations Research*, 12:568-581.

Crainic TG, Perboli G, Tadei R (2008): Extreme Point-based Heuristics for Three-dimensional Bin Packing. *INFORMS Journal on Computing*, 20:368–384.

Deif, I, Bodin, LD (1984): Extension of the Clarke and Wright Algorithm for Solving the Vehicle Routing Problem with Backhauling. A.E. Kidder (Ed.), Proceedings of the Babson Conference on Software Uses in Transportation and Logistics Management, Babson Park, MA, 75-96.

Fuellerer G, Doerner KF, Hartl R, Iori M (2010): Metaheuristics for Vehicle Routing Problems with Three-dimensional Loading Constraints. *European Journal of Operational Research*, 201:751–759.

Gajpal, Y, Abad, P (2009): Multi-ant Colony System (MACS) for a Vehicle Routing Problem with Backhauls. *European Journal of Operational Research*, 196:102-117.

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

Ghaziri, H, Osman, IH (2006) A Self-organizing Algorithm for the Vehicle Routing with Backhauls. *Journal of Scheduling*, 9:97-114.

Golden B, Raghavan S, Wasil E (eds) (2008): The Vehicle Routing Problem: Latest Advances and New Challenges. Operations research/computer science interfaces series, Vol. 43. Springer, Berlin.

Goetschalckx, M, Jacobs-Blecha, Ch (1989): The Vehicle Routing Problems with Backhauls. *European Journal of Operational Research*, 42:39-51.

Iori M, Martello S (2010): Routing Problems with Loading Constraints. *Top*, 18:4–27.

Hansen, P, Mladenovic, N (2001) Variable Neighborhood Search: Principles and Applications. *European Journal of Operational Research*, 130:449–467.

Hemmelmayr, VC, Doerner, KF, Hartl, RF (2009): A Variable Neighborhood Search Heuristic for Periodic Routing Problems. *European Journal of Operational Research*, 195:791-802.

Kritzinger S, Tricoire F, Doerner K, Hartl R (2011): A Unified Variable Neighborhood Search for Vehicle Routing Problems with Fixed Fleet Size. *Proceedings of the 9th Metaheuristics International Conference* (*MIC 2011*).

Kytöjoki, J., Nuortio, T, Bräysy, O, Gendreau, M (2007): An Efficient Variable Neighborhood Search Heuristic for Very Large Scale Vehicle Routing Problems. *Computers & Operations Research*, 34:2743 – 2757.

Lacomme, P; Toussaint, H; Duhamel, C (2013): A GRASP x ELS for the vehicle routing problem with basic three-dimensional loading constraints. Engineering Applications of Artificial Intelligence, 26: 1795:1810.

Lin, S. (1965): Computer Solutions of the Traveling Salesman Problem. *Bell Systems Technical Journal*, 44:2245-2269.

Mingozzi, A, Giorgi, S, Baldacci, R (1999): An Exact Method for the Vehicle Routing Problem with Backhauls. *Transportation Science,* 33:315-328.

Mladenovic, N, Hansen, P (1997) Variable Neighborhood Search. *Computers & Operations Research*, 24:1097-1100.

Moura A, Oliveira JF (2009): An Integrated Approach to Vehicle Routing and Container Loading Problems. *Operations Research Spectrum* 31:775–800.

Osman, IH, Wassan, N (2002): Reactive Tabu Search Meta-heuristic for the Vehicle Routing Problem with Backhauls. *Journal of Scheduling*, 5:263-285.

Parragh, SN, Doerner, KF, Hartl, RF (2008): A Survey on Pickup and Delivery Problems. Part I: Transportation between Customers and Depot. *Journal für Betriebswirtschaft*, 58:21-51.

Ropke, S, Pisinger, D (2006a): An Adaptive Large Neighborhood Search for the Pickup and Delivery Problem with Time Windows. *Transportation Science*, 40:455-472.

Ropke, S, Pisinger, D (2006b): A Unified Heuristic for a Large Class of Vehicle Routing Problems with Backhauls. *European Journal of Operational Research*, 171:750–775.

Ruan, Q, Zhang, Z, Miao, L, Shen, H (2013): A Hybrid Approach for the Vehicle Routing Problem with Three-dimensional Loading Constraints. *Computers & Operations Research,* 40:1579-1589.

Saremi, A, ElMekkawy, TY, Wang, GG (2007): Tuning the Parameters of a Memetic Algorithm to Solve Vehicle Routing Problem with Backhauls Using Design of Experiments. *International Journal of Operations Research,* 4:206−219.

Sprenger, R, Mönch, L (2012): A Methodology to Solve Large-scale Cooperative Transportation Planning Problems. *European Journal of Operational Research* 223:626–636.

Tarantilis CD, Zachariadis EE, Kiranoudis CT (2009): A Hybrid Metaheuristic Algorithm for the Integrated Vehicle Routing and Three-dimensional Container-loading Problem. *IEEE Transactions on Intelligent Transportation Systems,* 10:255–271.

Tavakkoli-Moghaddam, R, Saremi, A, Ziaee, M (2006): A Memetic Algorithm for a Vehicle Routing Problem with Backhauls. *Applied Mathematics and Computation*, 181:1049-1060.

Thangiah, SR, Potvin, J-Y, Sun, T (1997): Heuristic Approaches to Vehicle Routing with Backhauls and Time Windows. *Computers & Operations Research*, 23:1043-1057.

Toth, P, Vigo, D (1996): A Heuristic Algorithm for the Vehicle Routing Problem with Backhauls. In: L. Bianco und P. Toth (Hg.), Advanced Methods in Transportation Analysis, Springer, Berlin, 585-608.

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

Toth P, Vigo D (2002): The Vehicle Routing Problem. SIAM Monographs on Discrete Mathematics and Applications. Philadelphia, PA.

Vidal, T; Crainic, T G; Gendreau, M; Prins, C (2014): A unified solution framework for multi-attribute vehicle routing problems. *European Journal of Operational Research*, 234:658-673.

Wang, L, Guo, S, Chen, S, Zhu, W, Lim, A (2010): Two Natural Heuristics for 3D Packing with Practical Loading Constraints. *Computer Science,* Vol. 6230, 256-267.

Wassan, N (2007) Reactive Tabu Adaptive Memory Programming Search for the Vehicle Routing Problem with Backhauls. *Journal of the Operational Research Society*, 58:1630–1641.

Wisniewski, M, Ritt, M, Buriol, LS (2011): A Tabu Algorithm for the Capacitated Vehicle Routing Problem with Three-dimensional Loading Constraints. *Anais do XLIII Simpósio Brasileiro de Pesquisa Operacional*. Ubatuba, Brazil, 1502-1511.

Zachariadis, E, Kiranoudis, CT (2012): An Effective Local Search Approach for the Vehicle Routing Problem with Backhauls. *Expert Systems with Applications,* 39:3174-3184.

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. *Computers & Operations Research,* 39:2178-2195.

Verzeichnis der zuletzt erschienenen Informatik-Berichte

[353] Bauer, A., Dillhage, R., Hertling, P., Ko K.I., Rettinger, R.:
CCA 2009 Sixth International Conference on Computability and Complexity in Analysis

[354] Beierle, C., Kern-Isberner, G.:
Relational Approaches to Knowledge Representation and Learning

[355] Sakr, M.A., Güting, R.H.:
Spatiotemporal Pattern Queries

[356] Güting, R. H., Behr, T., Düntgen, C.:
SECONDO: A Platform for Moving Objects Database Research and for Publishing and Integrating Research Implementations

[357] Düntgen, C., Behr, T., Güting, R.H.:
Assessing Representations for Moving Object Histories

[358] Sakr, M.A., Güting, R.H.:
Group Spatiotemporal Pattern Queries

[359] Hartrumpf, S., Helbig, H., vor der Brück, T. , Eichhorn, C.:
SemDupl: Semantic Based Duplicate Identification

[360] Xu, J., Güting, R.H.:
A Generic Data Model for Moving Objects

[361] Beierle, C., Kern-Isberner, G.:
Evolving Knowledge in Theory and Application: 3rd Workshop on Dynamics of Knowledge and Belief, DKB 2011

[362] Xu, J., Güting, R.H.:
GMOBench: A Benchmark for Generic Moving Objects

[363] Finthammer, M.:
A Generalized Iterative Scaling Algorithm for Maximum Entropy Reasoning in Relational Probabilistic Conditional Logic Under Aggregation Semantics

[364] Güting, R.H., Behr, T., Düntgen, C.:
Book Chapter: Trajectory Databases

[365] Paul, A.; Rettinger, R.; Weihrauch, K.:
CCA 2012 Ninth International Conference on Computability and Complexity in Analysis (extended abstracts)

[366] Lu, J., Güting, R.H.:
Simple and Efficient Coupling of a Hadoop With a Database Engine

[367] Hoyrup, M., Ko, K., Rettinger, R., Zhong, N.:
CCA 2013 Tenth International Conference on Computability and Complexity in Analysis

[368] Beierle, C., Kern-Isberner, G.:
4th Workshop Dynamics of Knowledge and Belief (DKB-2013)

[369] Güting, R.H., Valdés, F., Damiani, M.L.:
Symbolic Trajectories