WORKING PAPER SERIES

A Hybrid Algorithm for the Vehicle Routing Problem with Pickup and Delivery and 3D Loading Constraints

Dirk Männel/Andreas Bortfeldt

Working Paper No. 15/2015

Impressum (§ 5 TMG)

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

Verantwortlich für diese Ausgabe:
Dirk Männel/Andreas Bortfeldt
Otto-von-Guericke-Universität Magdeburg
Fakultät für Wirtschaftswissenschaft
Postfach 4120
39016 Magdeburg
Germany

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

Bezug über den Herausgeber
ISSN 1615-4274

A Hybrid Algorithm for the Vehicle Routing Problem with Pickup and Delivery and 3D Loading Constraints

Dirk Männel[1], Andreas Bortfeldt[1*]
[1] Otto von Guericke University, Universitätsplatz 2, 39106 Magdeburg, Germany
andreas.bortfeldt@ovgu.de, dirk.maennel@gmx.de
[*] Corresponding author, phone: 0049 391 6711842

Abstract

In this paper, we extend the classical Pickup and Delivery Problem (PDP) to an integrated routing and three-dimensional loading problem, called PDP with 3D loading constraints (3L-PDP). A set of routes of minimum total length has to be determined such that each request is transported from a loading site to the corresponding unloading site. In the 3L-PDP, each request is given as a set of 3D rectangular items (boxes) and the vehicle capacity is replaced by a 3D loading space. We investigate which constraints will ensure that no *re*loading effort will occur, i.e. that no box is moved after loading and before unloading. A spectrum of 3L-PDP variants is introduced with different characteristics in terms of reloading effort. We propose a hybrid algorithm for solving the 3L-PDP consisting of a routing and a packing procedure. The routing procedure modifies a well-known large neighborhood search for the 1D-PDP. A tree search heuristic is responsible for packing boxes. Computational experiments were carried out using 54 newly proposed 3L-PDP benchmark instances.

Key words: Transportation, vehicle routing, pickup and delivery, 3D loading constraints.

1 Introduction

Routing vehicles and loading them with goods represent two major challenges in transportation logistics. Routing and loading problems have to be tackled as integrated problems if companies are interested in optimizing both the routing of vehicles and the corresponding loading of goods. Gendreau et al. (2006) first formulated and solved an integrated routing and loading problem, namely the capacitated vehicle routing problem (CVRP) with three-dimensional (3D) loading constraints (3L-CVRP). Contrasting to the classical CVRP, customer demands are represented as sets of parallel epipeds (called boxes) and the scalar capacity of a vehicle is replaced by a 3D rectangular loading space. This essential modification allows for a more detailed modeling of mixed cargo transportation by vehicles. Several packing constraints, e.g. concerning stacking of goods, can only be considered if customer demands are viewed as sets of 3D items. To ensure that calculated routes can actually be implemented, a 3D modeling of cargo and loading spaces is indispensable (see Bortfeldt and Homberger, 2013). Thus it seems to be desirable to model and solve further types of vehicle routing problems (VRPs) as integrated routing and 3D loading problems (3L-VRP).

This task is tackled here for the classical Pickup and Delivery Problem (PDP). In the classical PDP, we are given a set of transportation requests that have to be served by a fleet of homogeneous vehicles with a uniform 1D capacity. Each request is characterized by a 1D demand that has to be transported from a *specific* loading site (pickup point) to a *specific* unloading site (delivery point). Since we have a single pickup point and single delivery point per request, the classical PDP belongs to the one-to-one VRPs with pickup and delivery. A set of routes, each starting and ending at the single depot, has to be constructed in such a way that (i) each request is served at only one route and its loading site is visited before its unloading site; (ii) the capacity of a used vehicle is never exceeded by the set of loaded goods; (iii) the length of each route does not exceed a given limit; (iv) the number of routes does not exceed the given number of vehicles, and (v) the transportation cost, given by the total travel distance, is minimized.

To extend the classical PDP to an integrated routing and 3D loading problem, called hereafter PDP with three-dimensional loading constraints (3L-PDP), the demands are taken as sets of 3D rectangular items and the vehicles are equipped by a 3D rectangular loading space. As usual for the 3L-CVRP, we want to have a problem formulation that rules out any *re*loading effort. That is, the boxes should not be moved *after* loading and *before* unloading. In the 3L-CVRP this is guaranteed by the so-called Last-In-First-Out (LIFO) condition. It turns out that this constraint is not sufficient to eliminate any reloading effort for the 3L-PDP. Therefore, additional constraints are introduced for this purpose. This leads to a spectrum of 3L-PDP variants that are afterwards defined more formally.

A hybrid algorithm for solving the 3L-PDP is proposed that is composed of the modified large neighborhood search (LNS) algorithm by Ropke and Pisinger (2006) for the 1D-PDP and the tree search (TRS) algorithm for packing boxes by Bortfeldt (2012). The hybrid algorithm is tested by means of 54 newly introduced 3L-PDP benchmark instances with up to 100 requests.

The rest of the paper is organized as follows: Section 2 reviews the relevant literature. In Section 3 crucial features of the 3L-PDP are discussed and some variants of the 3L-PDP are formulated. Section 4 describes the hybrid algorithm, while in Section 5 numerical results of experiments are presented and analyzed. Conclusions are drawn and an outlook at further research is given in Section 6.

2 Related work

In our literature review, we will focus on recent work on the classical PDP with paired pickup and delivery points and on VRP with 3D loading constraints (3L-VRP). We refer the reader to Toth and Vigo (2014) for a comprehensive survey on vehicle routing.

2.1 Solution methods for the classical PDP

In pickup and delivery problems, goods or passengers are transported between customers or institutions. Following the classification schema by Parragh et al. (2008) the classical PDP is characterized by paired pickup and delivery points, i.e. each pickup point is generally associated with a special delivery point and vice versa. Moreover, the PDP deals with the transportation of goods; hence, no special constraints and objectives are involved concerning the (in)convenience of passengers as in dial-a-ride problems. A further distinction can be made with regard to the number of available vehicles and we will consider only the multi vehicle case, while the single vehicle case, representing an immediate extension of the Traveling Salesman Problem (TSP), is not considered here.

Mathematical models of the classical PDP or PDP with time windows (PDPTW) can be found, e.g. in Parragh et al. (2008) and in Toth and Vigo (2014). Recently published solution methods are surveyed in Berbeglia et al. (2007), Parragh et al. (2008) and Gendreau et al. (2008). The PDP is NP-hard, as it generalizes the TSP. Therefore, mainly classical heuristics and metaheuristics were proposed for solving the PDP. A representative sample of recent heuristics is listed in Table 1. For further details of the algorithms, the reader is referred to the references; some comments can be found in Parragh et al. (2008) and in Toth and Vigo (2014). For an introduction in metaheuristic approaches we refer to Gendreau and Potvin (2010).

Table 1: Sample of heuristics for the classical PDP.

Reference	Type of heuristic
Nanry and Barnes (2000)	Reactive tabu search
Li and Lim (2001)	Tabu embedded simulated annealing
Lim et al. (2002)	Squeeky wheel optimization
Pankratz (2005)	Grouping genetic algorithm
Lu and Dessouky (2006)	Construction heuristic
Bent and van Hentenryck (2006)	Hybrid algorithm: simulated annealing, large neighborhood search
Ropke and Pisinger (2006)	Adaptive large neighborhood search
Derigs and Döhmer (2008)	Indirect local search with greedy decoding
Nagata and Kobayashi (2010)	Guided ejection search

All solution methods listed in Table 1 are specified for the PDPTW, i.e. time windows are always considered. However, service times are only taken into account by Nanri and Barnes (2000), Li and Lim (2001) and by Ropke and Pisinger (2006). Almost all methods minimize the routing cost (total travel distance) and several methods do also minimize the number of routes. The multi depot case is only handled by Ropke and Pisinger (2006). Almost all methods of Table 1 assume that the vehicle fleet is homogeneous. The case of heterogeneous vehicles is dealt with by Xu et al. (2003) and by Ropke and Pisinger (2006). Most of the solution methods listed in Table 1 are evaluated by means of the benchmark instances proposed by Li and Lim (2001). Outstanding results especially for larger instances were achieved through the neighborhood search methods by Bent and van Hentenryck (2006) and by Ropke and Pisinger (2006), while the method of Li and Lim (2001) proved to be very successful for smaller instances.

An exact branch and cut algorithm for the PDPTW was proposed by Ropke et al. (2007), while Ropke and Cordeau (2009) specified a branch and cut and price algorithm. Baldacci et al. (2011) recently presented an exact algorithm based on a set-partitioning-like integer formulation. These exact PDPTW algorithms are capable of solving PDPTW instances with up to 500 requests; nevertheless, the numerical results reveal that heuristic approaches remain indispensable for large PDP instances.

2.2 Solution methods for vehicle routing problems with 3D loading constraints

Iori and Martello (2010, 2013) and Pollaris et al. (2015) 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 VRP with 3D loading constraints.

The 3L-CVRP was introduced by Gendreau et al. (2006) with five additional packing constraints frequently occurring in freight transportation. These include a last-in-first-out (LIFO) loading constraint, a weight constraint, an orientation constraint, a support constraint, and a stacking constraint (see Section 2 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. 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). Finally, Tao and Wang (2015) developed a tabu search procedure and hybridized it with an effective packing algorithm.

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. Zachariadis et al. (2012) consider a 3L-VRP with time windows where boxes are stacked on pallets, which in turn are loaded in vehicles.

Two hybrid algorithms for the 3L-VRP with backhauls were proposed by Bortfeldt et al. (2015). Both algorithms include a neighbourhood search algorithm for routing and a tree search algorithm for packing boxes.

The PDP with 3D loading constraints was up to now only tackled by Bartók and Imreh (2011). They specify a local search heuristic for solving a PDP variant that, however, does neglect the LIFO constraint and does not take into account other reasons for reloading effort (see Section 3.1). The 2L-PDP was covered by Malapert et al. (2008). They developed a constraint programming approach for the loading aspects of the problem but did not report numerical results. Ultimately, several basic types of 3L-VRP are not yet satisfactorily treated. This applies in particular for the 3L-PDP.

3 The 3L-PDP and some of its variants

Before giving a more formal definition of the 3L-PDP, we will discuss and illustrate crucial points of this problem.

3.1 Crucial features of the 3L-PDP

As in the classical PDP, a number of requests have to be transported from a pickup point to a delivery point by means of homogeneous vehicles. However, in the 3L-PDP, the demands consist of sets of boxes and they are sent in 3D loading spaces of the vehicles.

We assume that all vehicles are rear-loaded, i.e. the goods are loaded and unloaded at the rear exclusively by movements in length direction of the vehicle (cf. Figure 3). Lifting boxes or moving them in width direction is not permitted in the loading or unloading operation.

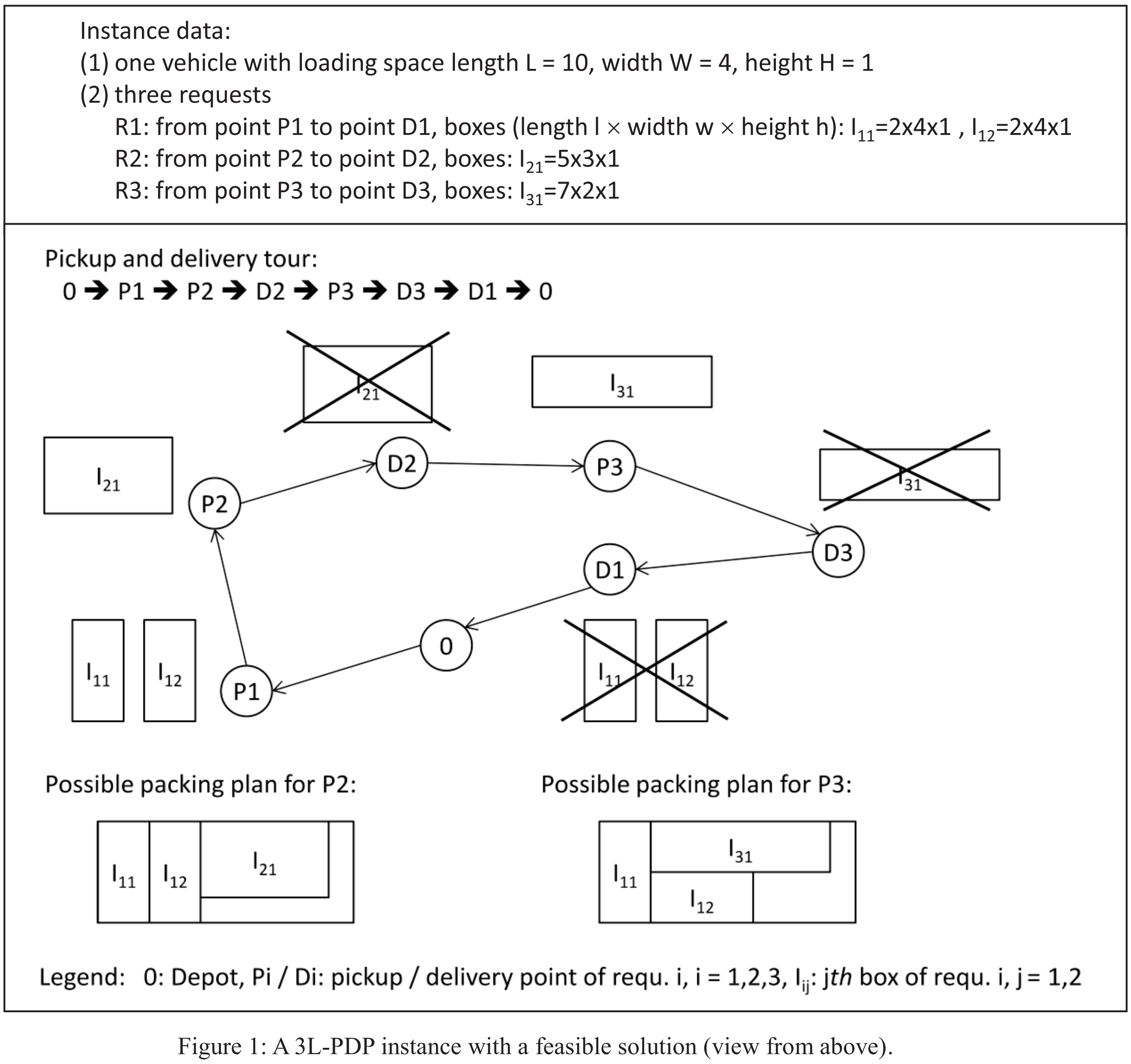
At the same time, we want to avoid any *re*loading effort, that is any temporary or permanent repositioning and rotating of boxes *after* loading and *before* unloading. There are different practical reasons to forbid reloading of goods during a pickup and delivery tour. Absence of manpower, tight working time, lack of equipment and shortage of space at customer sites are some of them. Moreover, the goods might be extra heavy, fragile or even hazardous.

Thus, the question arises, which conditions a pickup and delivery tour and the corresponding packing of boxes must observe to rule out any reloading effort.

The first condition is the request sequence (RS) constraint at delivery points that is well-known as last-in-first-out (LIFO) constraint from other integrated routing and loading problems, for example the 3L-CVRP. At a delivery point the RS constraint requires that between a box A to be unloaded and the rear of the vehicle no box B is situated that needs to be unloaded later. Likewise a box B to be unloaded later must not lie above box A. Otherwise box B has to be reloaded before box A can be unloaded by a pure movement in length direction.

As also pickup points occur in a pickup and delivery tour, we must have also a RS constraint to exclude reloading of goods at loading sites. At a pickup point the RS constraint requires that between the position of a box A just loaded and the rear of the vehicle or above box A no box B is situated that was loaded at an earlier pickup point. Again, otherwise a reloading of box B would be inevitable.

It is an essential feature of 3L-PDP that the RS constraint (for unloading and loading sites) is not sufficient to rule out any reloading effort. To understand this fact, we consider a simple 3L-PDP instance and a corresponding solution with one route and appropriate packing plans (see Figure 1).



Instance data:
(1) one vehicle with loading space length L = 10, width W = 4, height H = 1
(2) three requests
R1: from point P1 to point D1, boxes (length l × width w × height h): I_{11}=2x4x1 , I_{12}=2x4x1
R2: from point P2 to point D2, boxes: I_{21}=5x3x1
R3: from point P3 to point D3, boxes: I_{31}=7x2x1

Pickup and delivery tour:
0 ➔ P1 ➔ P2 ➔ D2 ➔ P3 ➔ D3 ➔ D1 ➔ 0

Possible packing plan for P2:

Possible packing plan for P3:

Legend: 0: Depot, Pi / Di: pickup / delivery point of requ. i, i = 1,2,3, I_{ij}: *jth* box of requ. i, j = 1,2

Figure 1: A 3L-PDP instance with a feasible solution (view from above).

It is evident that the route and the appropriate packing plans represent a feasible solution for the

3L-PDP instance given in Figure 1. In particular, the RS constraint in both variants is observed. Nevertheless, we have to state some reloading effort since box I_{12} of request 1 is rotated at the pickup point of request 3. Moreover, the given route could not be implemented if this reloading operation would not be done. Obviously, it is impossible to find two packing plans for the pickup points *P2* and *P3* such that the boxes of request 1 are located at the same positions in both plans. That is, the boxes of request 1 have to be reloaded at *P3* as otherwise the boxes of requests 1 and 3 could not be stowed together.

It is a specific attribute of the 3L-PDP that in a route boxes of a request *A* are generally transported and packed for a part of the route together with boxes of a request *B* and for another part of the route together with boxes of a request *C* (and not with the boxes of *B*) etc. In the above example the boxes of request 1 are transported first together with the boxes of request 2 and afterwards together with the boxes of request 3.

If packing plans are generated for the different partial routes in which the boxes of request *A* are "on board", these boxes will generally occupy different places. To exclude a change of placements without fail, i.e. to rule out a reloading effort, we have to introduce a new constraint, the reloading ban: The placement of any box (including the position of a reference corner and the spatial orientation of the box) must not be changed after the box was loaded and before the box is unloaded.

All in all, we can state that reloading effort for the 3L-PDP can only be avoided unerringly if the RS constraint (for unloading and loading sites) and the reloading ban (as above defined) are required at the same time.

However, to meet the reloading ban, we have to find routes for a 3L-PDP instance where each route is completed by a series of interrelated packing plans. In the above example, two interrelated plans are necessary: the first one must contain placements of boxes of requests 1 and 2, the second one must include placements of the boxes of requests 1 and 3. To meet the reloading ban, the placement of boxes of request 1 must be the same in the first and the second plan (making the plans interrelated). The specification of a packing procedure that is able to determine interrelated packing plans for greater routes in short running times presents a fairly difficult task. Therefore, we first look for a simplified 3L-PDP variant that allows us to avoid specifying a packing algorithm for interrelated packing plans.

Instead, we are going to eliminate any reloading effort within a route by means of routing patterns that ensure that the boxes of any request must not be stored together with the boxes of different requests in different partial routes.

The idea of a routing pattern for the 3L-PDP is quite simple. It consists of a series of sub-patterns. Each sub-pattern is a sequence of *m* (*m* ≥ 1) pickup points followed by the corresponding delivery points in inverse order. In Figure 2 two routing patterns and two routes that do *not* correspond to a routing pattern are shown.

Routing pattern with sub-patterns with maximal two requests:

$0 \Rightarrow P1 \rightarrow P2 \rightarrow D2 \rightarrow D1 \Rightarrow P3 \rightarrow P4 \rightarrow D4 \rightarrow D3 \Rightarrow 0$

Routing pattern with sub-patterns with maximal three requests:

$0 \Rightarrow P1 \rightarrow P2 \rightarrow P3 \rightarrow D3 \rightarrow D2 \rightarrow D1 \Rightarrow P4 \rightarrow P5 \rightarrow D5 \rightarrow D4 \Rightarrow 0$

Route that does not follow a routing pattern:

$0 \rightarrow P1 \rightarrow P2 \rightarrow D2 \rightarrow P3 \rightarrow D3 \rightarrow D1 \rightarrow 0$

Route that does not follow a routing pattern:

$0 \rightarrow P1 \rightarrow P2 \rightarrow D1 \rightarrow D2 \rightarrow 0$

Legend: 0: Depot, Pi / Di: pickup / delivery point of request i, i = 1,...,5, $\Rightarrow$: sub-pattern before is finished

Figure 2: Examples of routing patterns and routes that not correspond to routing patterns.

If a route follows a routing pattern, the loading space will become empty again each time after a sub-pattern is finished. Hence, the packing plans that are needed for subsequent sub-patterns (or partial routes) are independent of each other. The boxes of a request have to be stowed together only with boxes of requests of the same sub-pattern. Thus, only one packing plan per request is needed and there is no need to reload the boxes of any request.

If all routes of a solution of a 3L-PDP instance follow a routing pattern in the above sense, we will say that the independent partial routes (IPR) constraint holds.

We are now ready to specify a spectrum of five 3L-PDP variants (see Table 2). We always require the RS constraint at loading sites. The 3L-PDP variants are defined by means of the RS constraint for unloading sites, the reloading ban and the independent partial route constraint. For each variant and each constraint the entry is "y", if the constraint is required and "n" if not. In case the IPR condition and the RS constraint at loading sites is required, RS constraint at unloading sites and reloading ban are automatically satisfied (see Section 4.3); this is marked by entry "a". Provided none of the three defining constraints must be met, a high reloading effort is to be expected and the total travel distance will be very low. In case only the RS constraint at unloading sites holds the reloading effort will be medium while the total travel distance will be low. The same applies if only the reloading ban is required while the RS constraint is not to be observed. For the other variants the reloading effort is zero and the total travel distance is relatively large. Provided the reloading effort is ruled out by the independent partial route condition the total travel distance is higher as this constraint restricts the solution space stronger than the reloading ban.

Table 2: Five 3L-PDP variants (y: yes, n: no, a: automatically).

#	RS loading	RS unloading	Reloading ban	Independent partial routes	Reloading effort	Travel distance
1	y	n	n	n	high	very low
2	y	y	n	n	medium	low
3	y	n	y	n	medium	low
4	y	y	y	n	zero	medium
5	y	a	a	y	zero	high

As stressed above we will focus on the 3L-PDP without any reloading. Nevertheless, the first three 3L-PDP variants are also of interest. A comparison should consider not only the saving of total travel distance in variants 1 to 3 but also show the reloading effort of the first three variants. However, in this paper we will only consider the elimination of reloading effort by means of the independent partial routes condition (i.e. variants 3 and 4 are not considered here).

Further routing and packing constraints are taken into account. As in the 1D-PDP, we limit the number of routes by a given number of vehicles v_{max} (assuming that each vehicle performs only one route). Also the route length is limited explicitly since the capacity of vehicles does not force a limited route length in a pickup and delivery mode. As usual for 3L-VRP we extend the problem formulation by some packing constraints introduced by Gendreau et al. (2006), namely a weight constraint, an orientation constraint, a support constraint and a fragility constraint. This procedure is beneficial since it facilitates numerical comparisons with other 3L-VRPs. All six aforementioned constraints are included in all problem variants listed in Table 2.

3.2 Problem definition

Now we specify the 3L-PDP more formally. We are given n requests each consisting of a loading site i, an unloading site $n+i$ and a set I_i of goods that are to be transported from i to $n+i$ ($i = 1,\ldots,n$). There are v_{max} identical vehicles, originally located at the single depot (denoted by 0), with a rectangular loading space with length L, width W and height H. Let $V = \{0,1,\ldots,n,n+1,\ldots,2n\}$ be the set of all nodes, i.e. loading and unloading sites including the depot. Let E be a set of undirected edges (i,j) that connect all node pairs ($0 \leq i, j \leq 2n$, $i \neq j$) and let $G = (V, E)$ be the resulting graph. Let travel costs c_{ij} ($c_{ij} \geq 0$) be assigned to each edge (i,j) and let the travel costs be symmetric, i.e. $c_{ij} = c_{ji}$ ($0 \leq i, j \leq 2n$, $i \neq j$). Set I_i includes m_i rectangular pieces (boxes) I_{ik} and box I_{ik} has the length l_{ik}, the width w_{ik} and the height h_{ik} ($i = 1,\ldots,n$, $k = 1,\ldots,m_i$).

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 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 closest to the origin of the coordinates system; in addition, an orientation index o_{ik} indicates which of the possible spatial orientations is selected ($i = 1,...,n$, $k = 1,...,m_i$). 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 3 shows a loading space with placed boxes. Each vehicle is loaded and unloaded at the rear and empty at the beginning of a route.

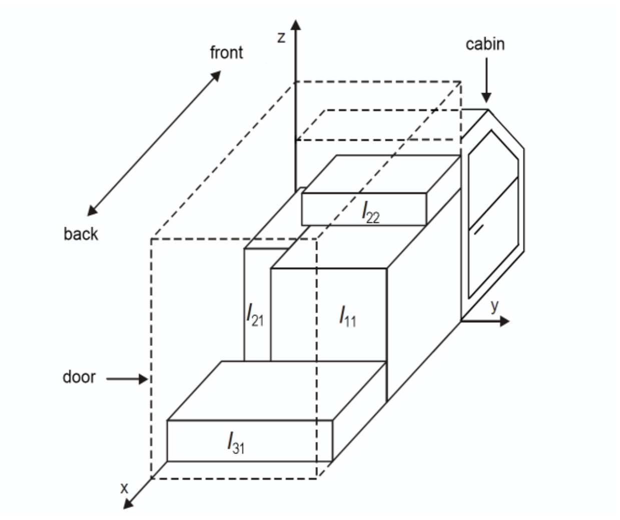


Figure 3: A loading space with placed boxes.

A feasible route R is a sequence of $2p$+2 nodes ($p \geq 1$) that starts and ends at the depot. R should include the loading and unloading sites of p different (among the n given) requests and each loading site must precede the unloading site of the same request. A solution of the 3L-PDP is a set of v sequences (R_l, $P_{l,1},\ldots,P_{l,2p_l}$), where R_l is a route and $P_{l,q}$ is a packing plan (l = 1,…,v, q = 1,…,2p_l, p_l denotes the number of requests of route l).

$P_{l,q}$ represents the packing pattern of route l after having visited its (q+1)*th* node, i.e. after some boxes were loaded or unloaded at the (q+1)*th* node of route l. To be feasible, a solution must fulfil the following three conditions: (F1) all routes R_l and packing plans $P_{l,q}$ are feasible (l = 1,…,v, q = 1,…,2p_l); (F2) the loading site and the unloading site of each request occurs once in one route R_l (l = 1,…,v); (F3) the packing plan $P_{l,q}$ for a route R_l and its (q+1)*th* node contains exactly placements for those boxes which are to be loaded but not (yet) to be unloaded at the first q+1 nodes of the route.

In addition, the following routing and packing constraints are to be satisfied optionally:

(C1) *Request sequence constraint (RS constraint)*: A packed box b of a certain request is said to be in unloading position if there is no packed box b' of another request between b and the rear of the vehicle or above box b (cf. Figure 3). *Loading requirement* (C1-l): If the (q+1)*th* node of route l is a loading site, then all boxes to be loaded there must be in unloading position in the packing plan $P_{l,q}$, i.e. after loading (l = 1,…,v, q = 1,…,2p_l). *Unloading requirement* (C1-u): If the (q+1)*th* node is an unloading site, then all boxes to be unloaded there must be in unloading position in the packing plan $P_{l,q-1}$, i.e. before unloading (l = 1,…,v, q = 1,…,2p_l). This constraint ensures that all boxes of a given request can be loaded or unloaded exclusively by movements parallel to the longitudinal axis of the loading space of a vehicle and without moving boxes of other requests.

(C2) *Reloading ban*: Each box I_{ik} of request i must not be moved *after* loading and *before* unloading (i = 1,…,n, k = 1,...,m_i). If the box I_{ik} is loaded at the (q+1)*th* node and unloaded at the (q'+1)*th* node of route l, its placement (x_{ik}, y_{ik}, z_{ik}, o_{ik}) must be the same in the packing plans $P_{l,q}$, $P_{l,q+1},\ldots,P_{l,q'-1}$ (i = 1,…,n, k = 1,...,m_i, l =1,…,v, $1 \leq q < q' \leq 2p_l$).

(C3) *Independent partial routes constraint*: Each route R_l follows a routing pattern, i.e. it consists of one or more sub-patterns (l = 1,…,v). A sub-pattern consists of a series of one or more loading sites (pickup points) followed by the corresponding unloading sites (delivery points) in inverse order.

(C4) *Weight constraint*: Each box I_{ik} has a positive weight d_{ik} (i = 1,...,n, k = 1,...,m_i) and the total weight of all boxes in a packing plan $P_{l,q}$ must not exceed a maximum load weight D (l = 1,...,v, q = 1,…,2p_l).

(C5) *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,...,n, k = 1,...,m_i).

(C6) *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.

(C7) *Stacking constraint*: A fragility attribute f_{ik} (i = 1,...,n, 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 ($f_{ik} = 0$).

(C8) *Route length constraint*: The total distance of a route must not exceed a specified maximum d_{max}. This constraint can also be understood as a route duration constraint if the vehicle velocity is set to a constant.

(C9) *Route number constraint*: The number of routes v must not exceed the number of vehicles v_{max}.

Finally, the 3L-PDP consists of determining a feasible solution that meets some of the constraints (C1) - (C9) and minimizes the total travel distance of all routes. More precisely, we consider the variants of 3L-PDP as specified in Table 1 (see above) and require constraints (C1) to (C3) in accordance to Table 1. The other constraints (C4) to (C9) are stipulated for each of the five variants of the 3L-PDP.

Figure 4 illustrates a simple 3L-PDP instance and the routes of a possible solution (while edges that do not belong to these routes are not shown.)

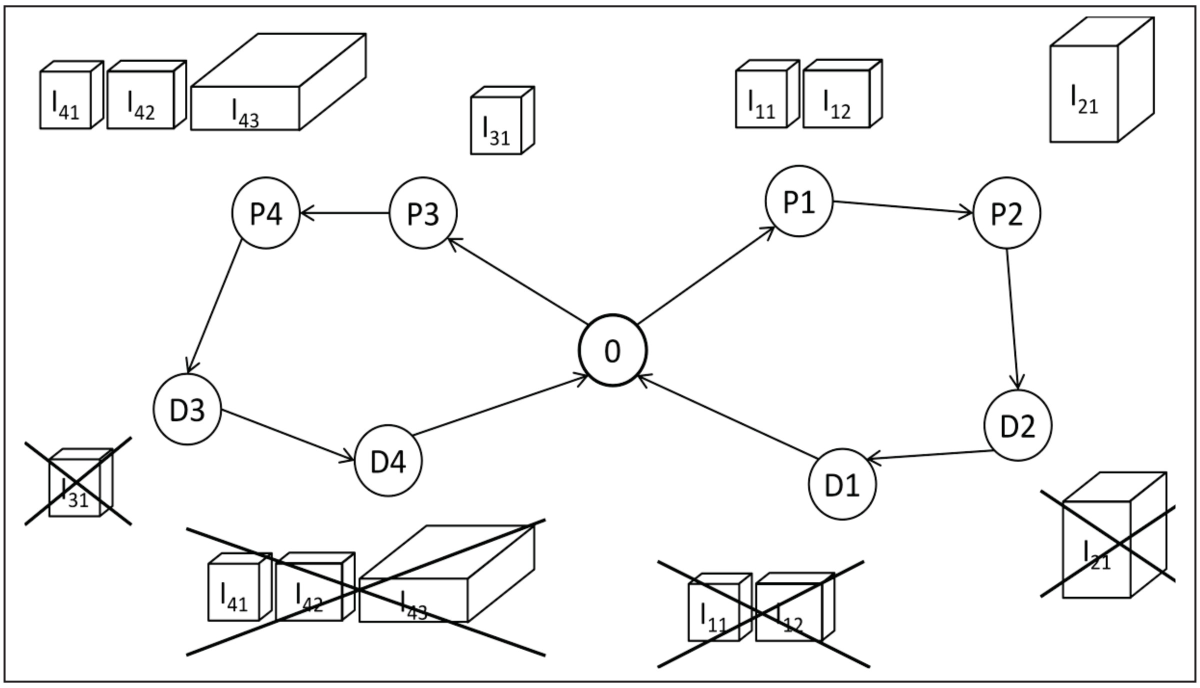


Figure 4: 3L-PDP instance with routes of a solution (legend see Fig. 2).

4 A hybrid algorithm for the 3L-PDP

In the sequel, we describe a hybrid algorithm for the 3L-PDP consisting of two separate procedures for routing and packing. The routing procedure is derived from the adaptive LNS heuristic for solving the PDPTW by Ropke and Pisinger (2006). Boxes are loaded into vehicles by the tree search 3D packing algorithm by Bortfeldt (2012). In the following description, emphasis is laid on the integration of routing and packing.

4.1 Routing procedure

The routing procedure is roughly outlined in Figure 5. First, an initial solution is constructed. Afterwards, an iterative neighborhood search is carried out until a time limit is exceeded. Within each iteration, a number ξ of requests to be removed and reinserted in the solution is selected randomly. Several removal and insertion heuristics are available. Among them, one removal and one insertion heuristic are selected randomly per iteration. 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.

```
3l_pdp_lns (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 requests to be removed $\xi$
        select removal heuristic Rh and insertion heuristic Ih
        determine next solution: $s_{next} := Ih(Rh(s_{curr}, \xi))$
        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 5: LNS-based routing algorithm for the 3L-PDP.

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 adaptive LNS, the selection probabilities for the removal and insertion heuristics are fix; i.e. a pure LNS is performed. Moreover, 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 requests are missing. To cope with this situation, the concept of a virtual request bank is used as in the original ALNS heuristic. The objective function is specified as the sum $f(s) = ttd(s) + M nmc(s)$, where s is a given solution, ttd stands for its total travel distance, nmc is the number of missing requests and M is a sufficiently large constant. By this definition, solutions with less missing requests are always preferred.

The removal and insertion heuristics are basically adopted from the original ALNS heuristic and briefly summarized in Table 3. Within the Shaw removal, the relatedness of requests is expressed by means of two factors, namely the locations of their pickup and delivery sites and the weights of their item sets. Hence, the relatedness of the two requests i and j is calculated by the blended index $r(i,j) = w_{r1}(c'_{ij} + c'_{i+n,j+n}) + w_{r2}|v'_i - v'_j|$, $1 \leq i < j \leq n$, where c'_{ij}, $c'_{i+n,j+n}$ and v'_i lie in the interval [0,1]; c'_{ij} ($c'_{i+n,j+n}$) denotes the normalized distance between the pickup points i and j (the delivery points $i+n$ and $j+n$); v'_i denotes the normalized weight of request i. The weights w_{rp} (p = 1, 2) allow for a different weighting of the distances and weights difference.

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.

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

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

The insertion heuristics are based on insertion moves and the concept of insertion cost. An insertion move is specified by four parameters: i denotes the inserted request ($1 \leq i \leq n$), k indicates the route in which i is inserted ($1 \leq k \leq v$), β and ε are the positions where the loading and unloading site of request i are inserted into route k ($1 \leq \beta \leq 2p(k)+1$, $2 \leq \varepsilon \leq 2p(k)+2$, $\beta < \varepsilon$); $p(k)$ is the number of requests that currently belong to route k. The insertion cost $\Delta f(i,k,\beta,\varepsilon)$ of an insertion move stands for the increase of the objective function value if the move is implemented. Only those insertion moves are admitted that do not violate the weight constraint (C4) and the route length constraint (C8). For a given request and a route k, having currently $p(k)$ requests, at most $(2p(k)+1)(2p(k)+2)/2$ feasible combinations for the index pair (β, ε) are available. Figure 6 shows two variants of inserting a request into an existing route.

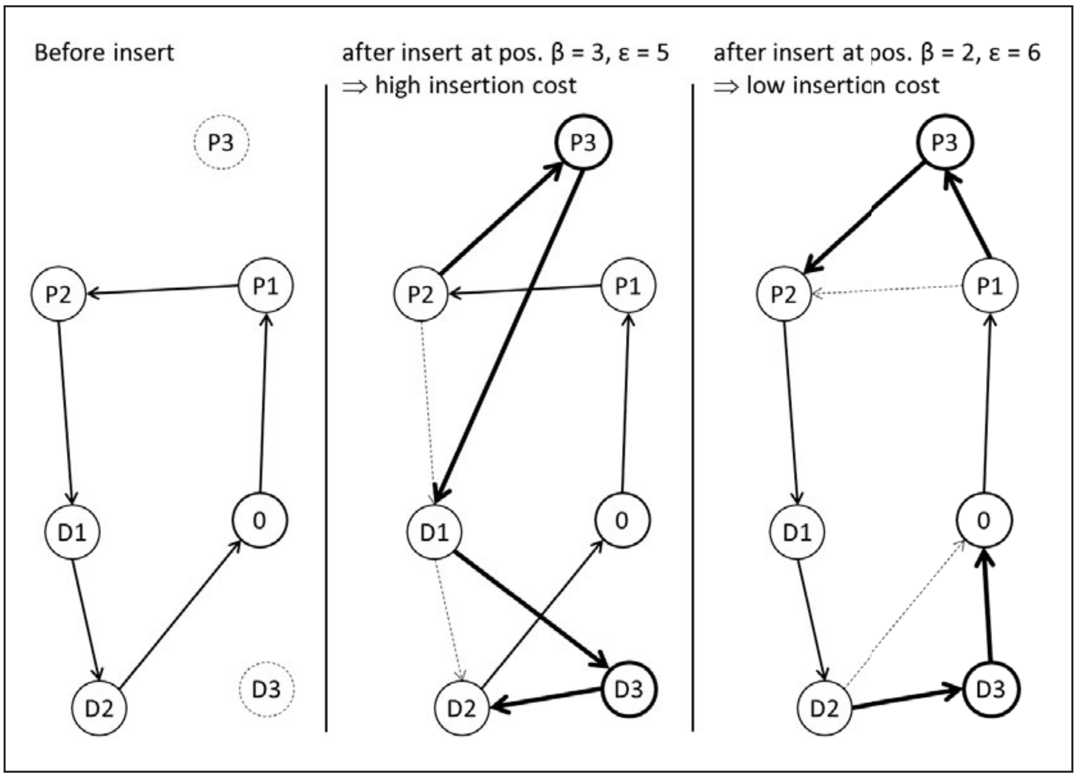


Figure 6: Insertion of a request into a route (bold lines: added edges, dotted lines: removed edges).

For the insertion variant $(\beta,\varepsilon) = (2,6)$ (on the right) the insertion cost $\Delta f(i,k,\beta,\varepsilon)$ results by adding together the distances $P1 \rightarrow P3$, $P3 \rightarrow P2$, $D2 \rightarrow D3$ and $D3 \rightarrow 0$ and subtracting then $P1 \rightarrow P2$ und $D2 \rightarrow 0$.

The insertion cost of a request i into a route k is specified as $\Delta f(i,k) = \min_{\beta,\varepsilon} \Delta f(i,k,\beta,\varepsilon)$, i. e. $\Delta f(i,k)$ is given by the cheapest insertion move of request i into route k. The insertion heuristics perform generally n iterations. The Greedy insertion Ih_G selects in each iteration the (request, route)-pair (i_0, k_0) for insertion which minimizes the insertion cost $(\Delta f(i^0, k^0) = \min_{i,k} \Delta f(i,k))$.

The Regret-2 insertion Ih_{R2} heuristic selects per iteration the request i_0 that maximizes the regret value $\rho_2(i) = \Delta f\big(i, k_2(i)\big) - \Delta f\big(i, k_1(i)\big)$; k_1 and k_2 $(k_1 \neq k_2)$ are those routes in which request i can be inserted with smallest and second smallest insertion cost $(\Delta f\big(i, k_1(i)\big) \leq \Delta f\big(i, k_2(i)\big) \leq \Delta f(i,k)$ $\forall\, k, k \neq k_1, k \neq k_2)$, respectively. Finally, we mention that the regret value for the Regret-3 insertion Ih_{R3} is calculated according $\rho_3(i) = \Delta f\big(i, k_2(i)\big) - \Delta f\big(i, k_1(i)\big) + \Delta f\big(i, k_3(i)\big) - \Delta f\big(i, k_1(i)\big)$.

The initial solution is constructed by means of the Regret-2 insertion heuristic starting with an empty solution.

4.2 Integration of routing and packing

To provide feasible packing plans for routes of solutions 3D packing checks are performed that are integrated in two parts of the routing procedure.

Let a new solution (s_{next}) be generated from an old one (s_{curr}) by means of a removal heuristic Rh and a insertion heuristic Ih according to $s_{next} := Ih(Rh(s_{curr}, \xi))$ and consider a route of s_{curr}. If Rh and Ih are applied to the route two cases can occur. In general, some requests (i.e. pairs of a pickup and delivery point) of the route are removed and some new requests are reinserted. It might also occur that only old requests are removed from the route without inserting new ones. In the former case, it will suffice to integrate packing checks in the insertion heuristic that is applied after the removal heuristic. In the latter case, it will be mostly possible to store the boxes of the remaining requests at all remaining sites of a route in a feasible way, too. Therefore, packing checks are integrated in the insertion heuristics exclusively (and not in removal heuristics) and these checks are called insertion packing checks.

However, sometimes the boxes of a given set can be stored in the loading space of a route 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 that is needed to provide sufficient support for another fragile box was removed. To cope with these cases, packing checks will also be applied to all routes of a solution s_{next} within the acceptance test of s_{next} (see Figure 5) and these checks are called acceptance packing checks. Especially all routes that did result earlier by a pure removing of requests are checked. If there is at least one site for which no feasible packing plan can be provided, the solution s_{next} will be discarded and the search continues with the last accepted solution. By this measure it is prevented that an accepted solution ever includes an infeasible route in terms of packing.

Subsequently, the integration of insertion packing checks is shown by means of the Greedy insertion heuristic.

In Figure 7 the Greedy insertion heuristic is shown in detail. It is based on the procedure select_best_insertions that performs the packing checks and is shown in Figure 8. The Greedy insertion heuristic takes an incomplete solution, a set of missing requests and the best solution so far as input values. In each loop cycle the best (minimum cost) insertion is determined, related to the requests still missing and implemented before the set of missing requests is updated. The procedure select_best_insertions is used to deliver the best insertion for a given request. In each cycle the number of still missing requests nmr_{wi} for which no feasible insertion was found at all is counted.

```
greedy_insertion (in: set of missing requests Rm, $s_{best}$, inout: solution s)
    repeat
        min_cost := $\infty$
        $nmr_{wi}$ := 0                                   // no. missing requests without insertion
        for all rq $\in$ Rm do
            $I_{best}$(rq) := select_best_insertions(s,rq,1) // set $I_{best}$(rq) receives best rq-insertion
            if |$I_{best}$(rq)| = 0 then $nmr_{wi}$ := $nmr_{wi}$ + 1    // request without insertion
            else
                if cost of best rq-insertion < min_cost then
                    $rq_{ins}$ := rq; min_cost := cost of best rq-insertion endif
            endif
        endfor
        if $nmr_{wi}$ > nmr($s_{best}$) then return endif          // no useful solution
        if $nmr_{wi}$ <|Rm| then update s by insertion in $I_{best}$($rq_{ins}$); Rm := Rm \ {$rq_{ins}$} endif
    until (|Rm| = 0 or $nmr_{wi}$ =|Rm|)
end.
```

Figure 7: Greedy insertion heuristic.

If in any cycle nmr_{wi} is greater than the number of missing requests $nmr(s_{best})$ 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 requests than the best solution so far or is even a feasible and complete solution.

The procedure select_best_insertions is organized in two parts. In the first part (*for*-loop) all potential insertions of a given request *rq* into any route of a given solution *s* are provided. Each insertion must be feasible (only) in terms of route length (C8), route number (C9) and weight (C4). In case of the 3L-PDP variants 2 or 5 the RS unloading constraint (C1-u) or the IPR constraint (C3) are checked here, too (to be explained later). 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, i.e. the insertion ins_{best} is applied to its route and the route is then checked in terms of the constraints (C1-l) and (C5) to (C7). 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}. 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, 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 technique, 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.

The procedure select_best_insertions is also used for the Regret-2 and Regret-3 insertion heuristics. In every cycle these heuristics retrieve the best two (three) insertions from select_best_insertions for all requests currently not contained in the (incomplete) solution to calculate the requests regret values. For each request, these two (three) insertions must belong to different routes. Finally in each cycle the request with maximal regret value is inserted into the solution, i.e. these requests' best insertion is implemented. For more details about the Regret-2 and Regret-3 insertion heuristic see Bortfeldt et al. (2015).

Our implementation of the LNS routing procedure can also be applied to the (1D) PDP. In this situation, the 3D packing test is omitted and only the route length (C8), route number (C9), and weight constraint (C4) are checked.

```
select_best_insertions (in: solution s, request rq, no. of required insertions $n_{ins}$,
                        out: set of best rq-insertions $I_{best}$)
    $I_{best} := \emptyset$; list of insertion candidates $I_{cand} := \emptyset$
    for all routes r of solution s do
        $I_{route}(r)$ := set of all 1D-feasible insertions of rq in route r
        sort $I_{route}(r)$ by ascending cost
        if $|I_{route}(r)| > 0$ then $I_{cand} := I_{cand} \cup \{I_{route}(r)(1)\}$ 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 $ins_{best} := I_{cand}(1)$; $I_{cand} := I_{cand} \setminus \{ins_{best}\}$
        perform 3D packing check of insertion $ins_{best}$, i.e. of the resulting route
        if 3D packing check of $ins_{best}$ successful then
            $I_{best} := I_{best} \cup \{ins_{best}\}$    // next best insertion found
        else r := route of $ins_{best}$
            $I_{route}(r) := I_{route}(r) \setminus \{ins_{best}\}$ // remove $ins_{best}$
            if $|I_{route}(r)| > 0$ then $I_{cand} := I_{cand} \cup \{I_{route}(r)(1)\}$ endif // add (new) first insertion
        endif
    endwhile
end.
```

Figure 8: Procedure select_best_insertions with packing check.

4.3 Packing checks

The insertion packing checks and the acceptance packing checks are now explained in detail.

Basically, for each route of a solution and each site visited in this route a feasible packing plan has to be provided. The plan must stow all boxes that are already loaded and not yet unloaded after the visit of this site. Now we ask whether existing feasible packing plans for selected sites of a route guarantee the existence of feasible packing plans for other sites. It turns out that this is the case at least if additional assumptions hold that are based on the required constraints of the 3L-PDP variants dealt with in this paper, i.e. the variants 1, 2 and 5 (see Table 2). By using these additional assumptions we want to reduce the effort spent for packing checks. We first deal with the 3L-PDP variants 2 and 5 and afterwards with variant 1.

We define a sequence of open pickup points (SOPP) as a sequence of pickup points within a route of a 3L-PDP solution having following characteristics: (i) the last point of the sequence is followed by a delivery point in the route; (ii) the sequence contains all and only pickup points of the route whose delivery points lie behind the last sequence point.

Let m_2 ($m_2 \geq 1$) be the number of consecutive pickup points lying at the end of the SOPP. Let m_1 ($m_1 \geq 0$) be the number of pickup points that are separated from the last m_2 pickup points by at least one delivery point. Then the sequence can be denoted as P_i, $i = 1,\ldots,m_1, m_1+1,\ldots,m_1+m_2$ (i.e. $P_{m_1+m_2}$ is the last point).

We say that a packing plan for pickup point $P_{m_1+m_2}$ of a SOPP satisfies the cumulative request sequence constraint for loading sites (CRS-l) if the following conditions hold: (i) there are no boxes of a request *j* (loaded at pickup point P_j) between a box of request *i* and the rear of the vehicle; (ii) there are no boxes of request *j* above a box of request *i* ($i, j = 1,\ldots,m_1+m_2, j < i$).

Proposition 1: Let a feasible plan for pickup point $P_{m_1+m_2}$ of a SOPP exist that meets the constraints (C1-l) and (C5) to (C7) and observes the CRS-l constraint. Then feasible packing plans observing constraints (C1-l) and (C5) to (C7) do also exist for pickup points P_i ($i = m_1+1,\ldots,m_1+m_2$-1).

Proof: Clearly, a packing plan for a pickup point that observes the CRS-l constraint also meets the RS constraint for loading sites (C1-l). If the boxes of the last request m_1+m_2 are removed, a packing plan for pickup point $P_{m_1+m_2\text{-}1}$ results that also meets the CRS-l constraint, hence the (C1-l) constraint. The plan for $P_{m_1+m_2\text{-}1}$ also satisfies support constraint (C6), as no boxes were removed that could serve for supporting boxes of requests 1 to m_1+m_2-1 (condition (ii) in CRS-l constraint definition). Constraints (C5) and (C7) as well as feasibility conditions (FP1) to (FP3) are trivially met in the plan for $P_{m_1+m_2\text{-}1}$ because they hold in the plan for $P_{m_1+m_2}$. For the pickup points P_i ($i = m_1+1,\ldots,m_1+m_2$-1), packing plans can be derived in a similar manner.

Now we introduce a further constraint with regard to routing called inverse delivery points sequence (IDPS) constraint. A route meets the IDPS constraint if the following condition holds: given any two requests i and j that are transported together at least between two consecutive points of a route; if the pickup point P_i lies before P_j, then the delivery point D_i lies behind D_j ($i,j = 1,\ldots,n$, $i < j$).

Proposition 2: Let a route be given that observes the IDPS constraint and a SOPP within the route. Let a packing plan for the (last) pickup point $P_{m_1+m_2}$ exist that is feasible, meets the constraints (C1-l) and (C5) to (C7) and satisfies the CRS-l constraint. Then feasible packing plans observing the constraints (C1-u) and (C5) to (C7) do also exist for the consecutive delivery points following pickup point $P_{m_1+m_2}$.

Proof: The points following $P_{m_1+m_2}$ must be the delivery points $D_{m_1+m_2}$, $D_{m_1+m_2\text{-}1},\ldots,$ $D_{m_1+m_2\text{-}\,m_3}$ ($0 \leq m_3 \leq m_2$) in this order since another set and another order of delivery points would contradict the IPDS constraint. The boxes of request m_1+m_2 are already in unloading position if the vehicle arrives in $D_{m_1+m_2}$. After these boxes were unloaded the boxes of request m_1+m_2-1 are in unloading position due to the CRS-l constraint, i.e. constraint (C1-u) is met in the plan for $D_{m_1+m_2}$. Constraints (C5) to (C7) and feasibility conditions (FP1) to (FP3) are verified for the plan for delivery point $D_{m_1+m_2}$ as in proof of Property 1. Feasible packing plans for further consecutive delivery points can be derived similarly.

The above considerations show that for 3L-PDP variants 2 and 5 it is sufficient to construct feasible packing plans for the last pickup points of all SOPPs of a given route. Feasible packing plans that observe the RS constraint (C1-l or C1-u) and constraints (C5) to (C7) can in this case be derived for all other pickup points and all delivery points of this route.

However, this claim holds only if two conditions are fulfilled. On the one hand, the constructed packing plans for the last pickup points of SOPPs must observe the CRS-l constraint being stronger than the (C1-l) constraint. On the other hand, the IDPS constraint for routes must be required. This constraint is included in the independent partial routes constraint (see Section 3.1); hence it holds automatically in problem variant 5. In problem variant 2 the IDPS constraint is substituted for the RS constraint for unloading sites (C1-u). This is possible as the RS constraint for loading sites (C1-l) and the IDPS constraint result in packing plans for delivery points that meet constraint (C1-u) as shown in Property 2 (this way also the entry "a" in line 5 and column "RS unloading" in Table 2 is justified).

Acceptance packing checks for problem variants 2 and 5 are carried out exactly as described before. An insertion packing check is performed each time another request is inserted into a route (see Section 3.2). Packing plans are then to be provided only for those last pickup points of SOPPs that lie between the inserted pickup point and the inserted corresponding delivery point since for the other parts of the route feasible packing plans are provided with the unmodified route of the former solution.

The procedure of packing checks for a route in 3L-PDP variants 2 and 5 is illustrated by two examples in Figure 9. In the first example (variant 2), boxes of some requests, e.g. 1, are loaded in multiple packing plans with possibly different placements. In the second example (variant 5) each request is stowed only one time due to the IPR constraint. Hence, there is no reloading effort at all.

In the 3L-PDP variant 1, the RS constraint for loading sites (C1-l) has to be observed. Hence, packing checks for pickup points are carried out as in problem variants 2 and 5. However, the RS constraint for unloading sites (C1-u) and a fortiori the stronger IDPS constraint no longer holds. Therefore, packing plans for delivery points can no longer be derived generally from packing plans for previous pickup points and their existence is no longer guaranteed.

Example for problem variant 2:

Given route

$0 \rightarrow P1 \rightarrow P2 \rightarrow D2 \rightarrow P3 \rightarrow P4 \rightarrow P5 \rightarrow D5 \rightarrow D4 \rightarrow P6 \rightarrow D6 \rightarrow D3 \rightarrow D1 \rightarrow 0$

Sequence of open pickup points	Packing plan incl. CRS-l constraint to be provided for site	Derived packing plans result for sites
1. $P1 \rightarrow P2$	P2	P1, D2
2. $P1 \rightarrow P3 \rightarrow P4 \rightarrow P5$	P5	P3, P4, D5, D6
3. $P1 \rightarrow P3 \rightarrow P6$	P6	D6, D3, D1

Example for problem variant 5:

Given route

$0 \rightarrow P1 \rightarrow P2 \rightarrow D2 \rightarrow D1 \rightarrow P3 \rightarrow P4 \rightarrow P5 \rightarrow D5 \rightarrow D4 \rightarrow D3 \rightarrow P6 \rightarrow D6 \rightarrow 0$

Sequence of open pickup points	Packing plan incl. CRS-l constraint to be provided for site	Derived packing plans result for sites
1. $P1 \rightarrow P2$	P2	P1, D2, D1
2. $P3 \rightarrow P4 \rightarrow P5$	P5	P3, P4, D5, D4, D3
3. P6	P6	D6

Legend: 0: Depot, Pi / Di: pickup / delivery point of request i, i = 1,...,6.

Figure 9: Packing checks in 3L-PDP variants 2 and 5.

Instead, feasible packing plans for all delivery sites of a given route must be provided separately. This is done in the following way:

(1) If a vehicle arrives at a delivery site, all boxes of the corresponding request, say *A*, are to be unloaded. Since the RS constraint (C1-u) is not required, some boxes of requests *B*, *C*, etc. may stand in the way of the *A*-boxes. All requests which were loaded after request *A* and are not already unloaded at the current delivery site are called blocking requests. For simplification it is assumed that all boxes of all blocking requests are to be temporarily unloaded.

(2) It can occur that there are no blocking requests at all, i.e. request *A* is the last in the loading sequence of requests. In this case the packing plan for the current delivery site results from the plan for the former site simply by removing the *A*-boxes. The resulting plan is feasible and observes the constraints (C5) to (C7) if the former plan has these characteristics (see Prop. 2).

(3) Otherwise a new feasible packing plan must be built for the current delivery site. The plan must contain placements for the loaded and not yet unloaded boxes (i.e. no longer placements of *A*-boxes).

(4) In particular, the requests that are temporarily to be unloaded must be considered for the new packing plan. We specify two policies for doing this. In the first policy (resulting in subvariant 1A), the original loading order of all requests is maintained. In the second policy (subvariant 1B), the original loading order is modified as the order of the temporarily unloaded requests is now chosen inverse to the order of corresponding delivery points. Clearly, the second loading order should result in less reloading effort at the following delivery sites.

(5) For the specified loading order, the packing heuristic generates a packing plan (if possible) in such a way that constraints (C5) to (C7) and the CRS-l constraint for the given loading order is fulfilled.

(6) For the requests that have to be temporarily unloaded, a reloading effort is to be calculated in any case. For the boxes of other requests, the placements in the packing plan for the current delivery site are compared with the placements in the packing plan of the previous site. Only for boxes with changed placements a reloading effort is calculated (including reloading effort for their blocking requests if necessary).

(7) Similar differences concerning the scope of checks exist between acceptance packing checks and insertion packing checks as in problem variants 2 and 5 (see above).

(8) The structure of input data of a packing check for a pickup point and for a delivery point is the same. In any case, the loading order of the relevant requests is needed as well as the box set per request.

Packing checks for problem variant 1 are illustrated by an example in Figure 10 that considers only delivery sites.

Example for problem variant 1:

Given route

$0 \rightarrow P1 \rightarrow P2 \rightarrow D1 \rightarrow P3 \rightarrow P4 \rightarrow P5 \rightarrow D3 \rightarrow D4 \rightarrow D5 \rightarrow D2 \rightarrow 0$

Delivery site	New packing plan to be provided	Relevant requests	Assumed blocking requests	Loading order
1. D1	yes	2	2	2
2. D3	yes	2,4,5	4,5	$2 \rightarrow 4 \rightarrow 5$ (1A) $2 \rightarrow 5 \rightarrow 4$ (1B)
3. D4	yes (1A) no (1B)	2,5	5	$2 \rightarrow 5$
4. D5	no			
5. D2	no			

Legend: 0: Depot, i: request no., Pi / Di: pickup / delivery point of request i, i = 1,...,5.

Figure 10: Packing checks for delivery sites in 3L-PDP variant 1.

4.4 Packing procedure

For a given loading sequence of requests and the corresponding sets of boxes, the packing procedure tries to determine a complete solution, i.e. a packing plan stowing all given boxes. The generated packing plan is feasible and observes the constraints (C5) to (C7) as well as the CRS-l constraint (including the (C1-l) constraint). A depth first search is carried out by means of the recursive procedure add_placement shown in Figure 11. 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 (incl. data concerning loading sequence) and the list *potentialPlacements* is filled by all feasible box placements in the lower left front corner of the loading space $L \times W \times H$ (cf. Figure 3).

The procedure add_placement checks first whether the current packing check can be aborted. 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 procedure add_placement is called again. To update the list *potentialPlacements,* all potential placements that can no longer be implemented are removed. 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 11: 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 CRS-l constraint. Placements of boxes are preferred that belong to earlier loaded requests and, therefore, have to be stowed nearer to the cabin. 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 loading sequences of requests that have ever been checked are collected in a cache to further accelerate the search. Whenever a sequence is tested, it is first searched in the cache. To speed up this search, for each request a separate table is established keeping the positions of all stored sequences including this request. Thus, a request sequence is searched by examining only the cache positions of its first request. 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 request sequences are avoided by this procedure.

5 Computational experiments

The computational experiments are organized in two parts. In the first part we examine the 1D variant of the hybrid algorithm (denoted by 1D-LNS) using the well-known PDP instances by Li and Lim (2001) with up to 100 requests. In the second part we test mainly the 3D variant of the algorithm by means of 54 new 3L-PDP instances with up to 100 requests and up to 300 boxes.

The packing procedure is coded in the C++ programming language using Visual Studio 2012 Express, while the LNS 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 i5-2500K (4.5 GHz, 16 GB RAM).

Afterwards, the new benchmark instances are introduced and the parameter setting is specified before the computational results are presented and analyzed.

5.1 Benchmark instances for 3L-PDP

To provide a sufficiently large set of 3L-PDP benchmark instances with different characteristics we generate the instances as follows:

- 30 instances with 50 requests, 18 instances with 75 and 6 instances with 100 requests are provided. The average number of boxes per request is two for half and three for the other half of the instances.
- Regarding the distribution of pickup and delivery sites of the requests, we distinguish the three variants "Random", "Mixed cluster" and "Pure cluster". In variant "Random" the sites are uniformly distributed in a rectangular section of the plane, while they are clustered in the other variants. In variant "Mixed cluster" individual clusters may contain pickup as well as delivery sites, while only sites of one sort can occur in an individual cluster of variant "Pure clusters". For each instance size, the same number of instances belongs to each distribution variant.
- Loading spaces and boxes are generated similarly to Gendreau et al. (2006). The dimensions of the uniform loading spaces of the vehicles are chosen as $L = 60$, $W = 25$ and $H = 30$ length units. The lengths l_{ik}, widths w_{ik} and heights h_{ik} of the boxes are drawn randomly from the intervals $[0.2 \cdot L, 0.6 \cdot L]$, $[0.2 \cdot W, 0.6 \cdot W]$ and $[0.2 \cdot H, 0.6 \cdot H]$, respectively ($i = 1,...,n$, $k = 1,...,m_i$). A box is characterized as fragile with the probability 0.25. The percentage a for the minimal supporting area was specified as 0.75.
- The weight capacity of the loading spaces was set to 45,000 weight units (being also the value of volume). The proportion of weight and volume is chosen three to one for the boxes of one third of the requests of an instance while for the boxes of the residual requests identical values are chosen for weights and volumes. Hereby the weight constraint (C4) does not become redundant.
- The maximal number of admitted vehicles v_{max} per instance is determined such that the algorithm can relatively easily find a feasible solution for problem variant 5. Since we expect shorter travel distances for the other problem variants, it should be possible to find feasible solutions for the other problem variants which respect the determined value of v_{max}, too.

The 54 new 3L-PDP instances are overviewed in Table 4; the figures in columns 2−8 are instance numbers. The instances will be provided at the website http://www.mansci.ovgu.de.

Table 4: Overview of the 54 new 3L-PDP benchmark instances.

Number of requests	2 Boxes per request on average			3 Boxes per request on average			Total
	Random	Mixed cluster	Pure cluster	Random	Mixed cluster	Pure cluster	
50	5	5	5	5	5	5	30
75	3	3	3	3	3	3	18
100	1	1	1	1	1	1	6

5.2 Parameter setting

The parameter setting for the experiments is specified in Table 5 and 6. The same parameterization of the routing procedures is used for all problem variants. All parameter values were determined based on limited computational experiments using a trial and error strategy.

Table 5: Parameter setting for the LNS routing procedure.

Parameter	Description	Value
r_{min}	lower bound of no. of removed customers	$0.04 \cdot n$
r_{max}	upper bound of no. of removed customers	$0.4 \cdot n$
w	start temperature control parameter	0.005
c	rate of geometrical cooling	0.9999
$p(Rh_R)$, $p(Rh_S)$	probability of Random / Shaw removal	0.3, 0.4
$p(Rh_W)$, $p(Rh_T)$	probability of Worst / Tour removal	0.1, 0.2
$p(Ih_G)$, $p(Ih_{R2})$, $p(Ih_{R3})$	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 6: Parameter setting for packing procedure.

Parameter	Description	Value
maxApCalls	Max. no. of calls of procedure add_placement	3000
maxBoxRankDiff	Max. tolerated rank difference of boxes	2
maxRefPoints	Max. number of admitted reference points	3

In Table 7 the maximum run time per instance and single run is shown. The computing time depends on the number of requests and the average box number per request. Lower time limits are specified if the new PDP instances are tested by 1D-LNS, i.e. as 1D-PDP instances.

Table 7: Computing time in minutes for experiments with new (3L-)PDP instances

Number of requests	1D-test (no packing)	2 Boxes per request on avg.	3 Boxes per request on avg.
50	1	2	5
75	2	4	10
100	4	8	20

For the 1D-LNS test by means of the instances by Li and Lim (2001) with 50 (100) requests, we allow a computing time of two (five) minutes.

5.3 Computational results for the one-dimensional PDP(TW)

The PDP instances by Li and Lim (2001) are actually PDPTW instances, i.e. the requests have time windows. To perform a comparison with other algorithms using the Li and Lim instances, we had to extend our hybrid algorithm to make it capable of taking time windows into account. Furthermore, most of the existing PDPTW solution procedures do minimize primarily the number of used vehicles (or routes) and the total travel distance is only the second objective criterion. We did adapt our hybrid algorithm also in this regard.

We tested 1D-LNS using the Li and Lim instances with 50 and 100 requests (100 and 200 customers, respectively) and compared the results of 1D-LNS with those of Ropke and Pisinger (2006) and the best known solutions. These were taken from the website Sintef (2015) and from Koning (2011).

Tables 8 and 9 present the best solutions achieved by 1D-LNS and by Ropke and Pisinger (2006) as well as the best known solutions as indicated in the above sources (*nv* is the number of used vehicles, *ttd* is the total travel distance). Best values are in bold; new best solutions are marked by an asterisk (*).

Table 8: Best solutions for Li-Lim-100 instances (50 requests)

Instance	1D-LNS		Ropke and Pisinger (2006)		Best known solution	
	nv	ttd	nv	ttd	nv	ttd
LC101	**10**	**828.94**	**10**	**828.94**	**10**	**828.94**
LC102	**10**	**828.94**	**10**	**828.94**	**10**	**828.94**
LC103	**9**	**1035.35**	**9**	**1035.35**	**9**	**1035.35**
LC104	**9**	**860.01**	**9**	**860.01**	**9**	**860.01**
LC105	**10**	**828.94**	**10**	**828.94**	**10**	**828.94**
LC106	**10**	**828.94**	**10**	**828.94**	**10**	**828.94**
LC107	**10**	**828.94**	**10**	**828.94**	**10**	**828.94**
LC108	**10**	**826.44**	**10**	**826.44**	**10**	**826.44**
LC109	**9**	**1000.60**	**9**	**1000.60**	**9**	**1000.60**
LC201	**3**	**591.56**	**3**	**591.56**	**3**	**591.56**
LC202	**3**	**591.56**	**3**	**591.56**	**3**	**591.56**
LC203	3	591.17	3	591.17	**3**	**585.56**
LC204	**3**	**590.60**	**3**	**590.60**	**3**	**590.60**
LC205	**3**	**588.88**	**3**	**588.88**	**3**	**588.88**
LC206	**3**	**588.49**	**3**	**588.49**	**3**	**588.49**
LC207	**3**	**588.29**	**3**	**588.29**	**3**	**588.29**
LC208	**3**	**588.32**	**3**	**588.32**	**3**	**588.32**
LR101	**19**	**1650.80**	**19**	**1650.80**	**19**	**1650.80**
LR102	**17**	**1487.57**	**17**	**1487.57**	**17**	**1487.57**
LR103	**13**	**1292.68**	**13**	**1292.68**	**13**	**1292.68**
LR104	**9**	**1013.39**	**9**	**1013.39**	**9**	**1013.39**
LR105	**14**	**1377.11**	**14**	**1377.11**	**14**	**1377.11**
LR106	**12**	**1252.62**	**12**	**1252.62**	**12**	**1252.62**
LR107	**10**	**1111.31**	**10**	**1111.31**	**10**	**1111.31**
LR108	**9**	**968.97**	**9**	**968.97**	**9**	**968.97**
LR109	**11**	**1208.97**	**11**	**1208.96**	**11**	**1208.96**
LR110	**10**	**1159.35**	**10**	**1159.35**	**10**	**1159.35**
LR111	**10**	**1108.90**	**10**	**1108.90**	**10**	**1108.90**
LR112	**9**	**1003.77**	**9**	**1003.77**	**9**	**1003.77**
LR201	**4**	**1253.23**	**4**	**1253.23**	**4**	**1253.23**
LR202	**3**	**1197.67**	**3**	**1197.67**	**3**	**1197.67**
LR203	**3**	**949.40**	**3**	**949.40**	**3**	**949.40**
LR204	**2**	**849.05**	**2**	**849.05**	**2**	**849.05**
LR205	**3**	**1054.02**	**3**	**1054.02**	**3**	**1054.02**
LR206	**3**	**931.63**	**3**	**931.63**	**3**	**931.63**
LR207	**2**	**903.06**	**2**	**903.06**	**2**	**903.06**
LR208	**2**	**734.85**	**2**	**734.85**	**2**	**734.85**
LR209	**3**	**930.59**	**3**	**930.59**	**3**	**930.59**
LR210	**3**	**964.22**	**3**	**964.22**	**3**	**964.22**
LR211	**2**	**911.52**	**2**	**911.52**	**2**	**911.52**
LRC101	**14**	**1708.80**	**14**	**1708.80**	**14**	**1708.80**
LRC102	**12**	**1558.07**	**12**	**1558.07**	**12**	**1558.07**
LRC103	**11**	**1258.74**	**11**	**1258.74**	**11**	**1258.74**
LRC104	**10**	**1128.40**	**10**	**1128.40**	**10**	**1128.40**
LRC105	**13**	**1637.62**	**13**	**1637.62**	**13**	**1637.62**
LRC106	**11**	**1424.73**	**11**	**1424.73**	**11**	**1424.73**
xLRC107	**11**	**1230.15**	**11**	**1230.14**	**11**	**1230.14**
LC101	**10**	**828.94**	**10**	**828.94**	**10**	**828.94**
LC102	**10**	**828.94**	**10**	**828.94**	**10**	**828.94**
LC103	**9**	**1035.35**	**9**	**1035.35**	**9**	**1035.35**
LC104	**9**	**860.01**	**9**	**860.01**	**9**	**860.01**
LC105	**10**	**828.94**	**10**	**828.94**	**10**	**828.94**
LC106	**10**	**828.94**	**10**	**828.94**	**10**	**828.94**
LC107	**10**	**828.94**	**10**	**828.94**	**10**	**828.94**
LC108	**10**	**826.44**	**10**	**826.44**	**10**	**826.44**
LC109	**9**	**1000.60**	**9**	**1000.60**	**9**	**1000.60**

For 55 of 56 Li-Lim-100 instances 1D-LNS finds the best known solution and reaches exactly the same results as ALNS by Ropke and Pisinger (2006) for all 56 instances. For the Li-Lim-200 instances 1D-LNS performs slightly worse than ALNS. Both algorithms achieve the minimal known number of vehicles for 59 of 60 instances. The average gap of the total travel distance amounts to 0.24 % for 1D-

LNS and to 0.09 % for ALNS.

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 ten runs) and *ttd-best-known* is the best known travel distance for the given instance. Moreover, 1D-LNS achieves new best solutions for ten instances. Here we have to make the restrictive remark that we only claim to achieve (old or new) best solutions with regard to the above mentioned references.

Table 9: Best solution for Li-Lim-200 instances (100 requests)

Instance	1D-LNS		Ropke and Pisinger (2006)		Best known solution	
	nvt	ttd	nv	ttd	nv	ttd
LC1_2_1	**20**	**2704.57**	**20**	**2704.57**	**20**	**2704.57**
LC1_2_2	**19**	**2764.56**	**19**	**2764.56**	**19**	**2764.56**
LC1_2_3	**17**	**3127.78***	17	3128.61	17	3128.61
LC1_2_4	**17**	**2693.41**	**17**	**2693.41**	**17**	**2693.41**
LC1_2_5	**20**	**2702.05**	**20**	**2702.05**	**20**	**2702.05**
LC1_2_6	**20**	**2701.04**	**20**	**2701.04**	**20**	**2701.04**
LC1_2_7	**20**	**2701.04**	**20**	**2701.04**	**20**	**2701.04**
LC1_2_8	**20**	**2689.83**	**20**	**2689.83**	**20**	**2689.83**
LC1_2_9	**18**	**2724.24**	**18**	**2724.24**	**18**	**2724.24**
LC1_2_10	**17**	**2942.13***	17	2943.49	17	2943.49
LC2_2_1	**6**	**1931.44**	**6**	**1931.44**	**6**	**1931.44**
LC2_2_2	**6**	**1881.40**	**6**	**1881.40**	**6**	**1881.40**
LC2_2_3	**6**	**1844.33**	**6**	**1844.33**	**6**	**1844.33**
LC2_2_4	6	1767.82	**6**	**1767.12**	**6**	**1767.12**
LC2_2_5	**6**	**1891.21**	**6**	**1891.21**	**6**	**1891.21**
LC2_2_6	**6**	**1857.78**	**6**	**1857.78**	**6**	**1857.78**
LC2_2_7	**6**	**1850.13**	**6**	**1850.13**	**6**	**1850.13**
LC2_2_8	**6**	**1824.34**	**6**	**1824.34**	**6**	**1824.34**
LC2_2_9	**6**	**1854.21**	**6**	**1854.21**	**6**	**1854.21**
LC2_2_10	**6**	**1817.45**	**6**	**1817.45**	**6**	**1817.45**
LR1_2_1	**20**	**4819.12**	**20**	**4819.12**	**20**	**4819.12**
LR1_2_2	**17**	**4621.21**	**17**	**4621.21**	**17**	**4621.21**
LR1_2_3	**15**	**3612.64**	**15**	**3612.64**	**15**	**3612.64**
LR1_2_4	**10**	**3031.20***	10	3037.38	10	3037.38
LR1_2_5	**16**	**4760.18**	**16**	**4760.18**	**16**	**4760.18**
LR1_2_6	**14**	**4175.16**	14	4178.24	**14**	**4175.16**
LR1_2_7	**12**	**3543.56***	12	3550.61	12	3550.61
LR1_2_8	9	2791.67	**9**	**2784.53**	**9**	**2784.53**
LR1_2_9	**14**	**4343.86***	14	4354.66	14	4354.66
LR1_2_10	**11**	**3695.84***	11	3714.16	11	3714.16
LR2_2_1	**5**	**4073.10**	**5**	**4073.10**	**5**	**4073.10**
LR2_2_2	4	3796.81	**4**	**3796.00**	**4**	**3796.00**
LR2_2_3	**4**	**3098.36**	**4**	**3098.36**	**4**	**3098.36**
LR2_2_4	3	2500.04	**3**	**2486.14**	**3**	**2486.14**
LR2_2_5	**4**	**3438.39**	**4**	**3438.39**	**4**	**3438.39**
LR2_2_6	**4**	**3201.54**	**4**	**3201.54**	**4**	**3201.54**
LR2_2_7	3	3152.52	**3**	**3135.05**	**3**	**3135.05**
LR2_2_8	2	2582.35	**2**	**2555.40**	**2**	**2555.40**
LR2_2_9	3	4054.50	**3**	**3930.49**	**3**	**3930.49**
LR2_2_10	**3**	**3286.95***	3	3344.08	3	3323.37
LRC1_2_1	**19**	**3606.06**	**19**	**3606.06**	**19**	**3606.06**
LRC1_2_2	15	3681.07	15	3674.80	**15**	**3671.02**
LRC1_2_3	**13**	**3154.92***	13	3178.17	13	3161.75
LRC1_2_4	**10**	**2631.82**	**10**	**2631.82**	**10**	**2631.82**
LRC1_2_5	**16**	**3715.81**	**16**	**3715.81**	**16**	**3715.81**
LRC1_2_6	17	3368.66	17	3368.66	**16**	**3572.16**
LRC1_2_7	14	3738.47	**14**	**3668.39**	**14**	**3668.39**
LRC1_2_8	13	3167.23	13	3174.55	**13**	**3146.70**
LRC1_2_9	13	3303.18	13	3226.72	**13**	**3157.34**
LRC1_2_10	12	2951.90	**12**	**2951.29**	**12**	**2951.29**
LRC2_2_1	6	3632.37	6	3605.40	**6**	**3595.18**
LRC2_2_2	**5**	**3182.62***	5	3327.18	5	3327.18
LRC2_2_3	**4**	**2914.82***	4	2938.28	4	2938.28
LRC2_2_4	3	3038.16	**3**	**2887.97**	**3**	**2887.97**
LRC2_2_5	5	2777.23	**5**	**2776.93**	**5**	**2776.93**
LRC2_2_6	**5**	**2707.96**	**5**	**2707.96**	**5**	**2707.96**
LRC2_2_7	4	3067.91	4	3056.09	**4**	**3044.40**
LRC2_2_8	4	2401.17	**4**	**2399.95**	**4**	**2399.95**
LRC2_2_9	4	2208.72	**4**	**2208.49**	**4**	**2208.49**
LRC2_2_10	3	2600.41	**3**	**2550.56**	**3**	**2550.56**

As the results indicate, our LNS implementation reaches nearly the solution quality of the best available methods for the PDPTW. In particular, a high solution quality was achieved although the LNS has been rather simplified compared to the original procedure by Ropke and Pisinger (2006).

5.4 Computational results for the 3L-PDP

The detailed results for the 3L-PDP instances regarding total travel distance (*ttd*) are presented in Table 10. In the leftmost column the instance names are listed. The next column shows the total travel distances for the 1D test for which only the weight constraint (C4), and the routing constraints (C8) and (C9) are considered. In the following eight columns the total travel distances and the gaps are indicated for the 3L-PDP variants 1A, 1B, 2 and 5 (see Table 2). All presented total travel distances are mean values over five runs. The corresponding gaps are calculated as (*ttd – ttd-1D)*/*ttd-1D**100 (%). In the last line of Table 10 the gap values of the 3L-PDP variants are averaged over the 54 instances.

Summarizing the results, we can state that the travel distances increase significantly if the 3L-PDP instances are solved instead of the corresponding 1D-PDP instances. For the problem variants 1A and 1B, the total travel distances grow on average by 11.31% and by 11.66%, respectively, compared to the 1D case. For the problem variants 2 and 5, the mean increase is even higher and amounts to 20.94 and 25.25%, respectively.

In Table 11 the reloading effort for the 3L-PDP variants 1A, 1B and 2 is indicated. Since the reloading effort is zero for problem variant 5, this variant does not occur in Table 11 (cf. Table 2).

The reloading effort needed for a 3L-PDP instance is primarily given as reloading quantity, i.e. as the weight of all boxes that are reloaded. If a box is reloaded, say, three times the weight of the box is counted three times. Thus it may occur that the reloading quantity exceeds the total weight of the boxes.

Table 11 is organized as follows. The first column includes the instance names and the second column shows the total weight of all requests per instance (cargo weight). In the following six columns the reloading quantities for the relevant 3L-PDP variants are given as absolute values (in weight units) and as percentages of the cargo weight. The results per instance are, again, averaged over five runs. In the last line of Table 11 the percentaged reloading quantities are averaged over the 54 instances.

The reloading quantity of problem variant 2, caused by the missing reloading ban, is only moderate and amounts to 15.82% of the cargo weight on average. For problem variants 1A and 1B the mean reloading quantity is much higher (101.23% and 89.83%, respectively), since not only the reloading ban is missing but the RS constraint (C1-u) for unloading sites is not required, either. The difference of 11.40%-points between the subvariants of problem variant 1 is also plausible, since in subvariant 1B the reloading quantity is partially saved by a better loading order of the temporarily unloaded requests.

Table 12 summarizes the results regarding total travel distance and reloading effort. For each 3L-PDP variant, the total travel distance is here given as percentage of the travel distance in the 1D-case while the reloading quantities are again indicated as percentages of the cargo weight. All presented values are averaged over the five runs per instance and over the 54 3L-PDP instances.

The results clearly show that a 3D modeling of pickup and delivery problems leads to a significant increase of the travel distances in the 3D case and is, therefore, generally more realistic. The indicated figures for the 3L-PDP variants correspond very well with the expected differences between those variants regarding travel distances and reloading effort as shown in Table 2. The main result is the clear tradeoff between travel distances and reloading effort indicating that a saving of travel distance has to be "paid" with an additional portion of reloading effort.

Table 10: Results (travel distances) for different variants of 3L-PDP

Instance	1D-Test	Variant 1A		Variant 1B		Variant 2		Variant 5	
	ttd	ttd	gap (%)	ttd	gap (%)	ttd	gap (%)	ttd	gap (%)
50_RAND_2_1	1362.26	1455.02	6.81	1454.24	6.75	1625.45	19.32	1739.10	27.66
50_RAND_2_2	1181.42	1314.65	11.28	1318.04	11.56	1505.18	27.40	1586.44	34.28
50_RAND_2_3	1234.98	1332.90	7.93	1334.59	8.07	1569.91	27.12	1666.28	34.92
50_RAND_2_4	1246.39	1366.41	9.63	1367.16	9.69	1538.08	23.40	1585.81	27.23
50_RAND_2_5	1276.39	1333.79	4.50	1354.64	6.13	1540.72	20.71	1594.02	24.88
50_CLUS_2_1	888.14	987.54	11.19	979.58	10.30	1055.68	18.86	1121.46	26.27
50_CLUS_2_2	852.82	931.02	9.17	927.90	8.80	1036.49	21.54	1109.68	30.12
50_CLUS_2_3	921.30	989.54	7.41	1011.51	9.79	1100.10	19.41	1151.12	24.94
50_CLUS_2_4	1031.28	1121.43	8.74	1117.70	8.38	1218.93	18.20	1275.24	23.66
50_CLUS_2_5	1132.02	1237.05	9.28	1231.75	8.81	1306.87	15.45	1379.22	21.84
50_CPCD_2_1	1102.82	1267.67	14.95	1259.81	14.24	1305.88	18.41	1378.32	24.98
50_CPCD_2_2	1039.63	1186.62	14.14	1168.05	12.35	1242.83	19.54	1260.34	21.23
50_CPCD_2_3	996.64	1135.87	13.97	1140.37	14.42	1196.29	20.03	1234.57	23.87
50_CPCD_2_4	1128.00	1257.23	11.46	1254.35	11.20	1307.38	15.90	1332.65	18.14
50_CPCD_2_5	1237.93	1370.15	10.68	1391.59	12.41	1430.77	15.58	1460.59	17.99
50_RAND_3_1	1359.03	1460.43	7.46	1467.18	7.96	1618.03	19.06	1715.25	26.21
50_RAND_3_2	1184.75	1306.15	10.25	1326.86	11.99	1463.17	23.50	1581.58	33.49
50_RAND_3_3	1251.30	1340.69	7.14	1345.13	7.50	1566.62	25.20	1661.71	32.80
50_RAND_3_4	1266.16	1347.32	6.41	1381.39	9.10	1532.35	21.02	1564.35	23.55
50_RAND_3_5	1281.81	1359.31	6.05	1357.90	5.94	1534.21	19.69	1588.92	23.96
50_CLUS_3_1	889.57	967.53	8.76	983.86	10.60	1007.47	13.25	1050.41	18.08
50_CLUS_3_2	854.41	920.50	7.74	924.67	8.22	1029.19	20.46	1099.70	28.71
50_CLUS_3_3	926.75	989.50	6.77	997.03	7.58	1081.18	16.66	1123.87	21.27
50_CLUS_3_4	1026.55	1103.63	7.51	1091.72	6.35	1204.17	17.30	1252.35	22.00
50_CLUS_3_5	1137.96	1240.91	9.05	1228.89	7.99	1286.80	13.08	1323.39	16.29
50_CPCD_3_1	1097.01	1307.50	19.19	1298.45	18.36	1322.84	20.59	1342.60	22.39
50_CPCD_3_2	1048.17	1204.86	14.95	1222.38	16.62	1239.50	18.25	1245.59	18.83
50_CPCD_3_3	998.97	1149.57	15.08	1140.22	14.14	1211.05	21.23	1252.47	25.38
50_CPCD_3_4	1135.69	1284.56	13.11	1283.16	12.99	1304.17	14.84	1313.58	15.66
50_CPCD_3_5	1237.80	1378.63	11.38	1403.03	13.35	1429.27	15.47	1450.47	17.18
75_RAND_2_1	1701.32	1867.59	9.77	1876.01	10.27	2053.82	20.72	2118.13	24.50
75_RAND_2_2	1562.31	1762.19	12.79	1762.22	12.80	2053.95	31.47	2130.80	36.39
75_RAND_2_3	1653.16	1793.10	8.46	1839.14	11.25	2106.54	27.42	2188.07	32.36
75_CLUS_2_1	1180.71	1315.88	11.45	1315.95	11.45	1390.50	17.77	1469.57	24.46
75_CLUS_2_2	1136.71	1277.16	12.36	1252.09	10.15	1378.23	21.25	1425.73	25.43
75_CLUS_2_3	1163.14	1343.67	15.52	1342.37	15.41	1440.17	23.82	1495.71	28.59
75_CPCD_2_1	1803.87	1995.94	10.65	2012.83	11.58	2177.45	20.71	2237.26	24.03
75_CPCD_2_2	1792.68	2021.17	12.75	2033.39	13.43	2188.77	22.10	2226.24	24.19
75_CPCD_2_3	1879.48	2106.77	12.09	2141.70	13.95	2233.85	18.85	2283.54	21.50
75_RAND_3_1	1698.67	1896.49	11.65	1867.50	9.94	2055.42	21.00	2146.59	26.37
75_RAND_3_2	1562.97	1727.39	10.52	1745.46	11.68	1968.19	25.93	2081.42	33.17
75_RAND_3_3	1654.16	1789.52	8.18	1785.70	7.95	2068.74	25.06	2146.31	29.75
75_CLUS_3_1	1178.62	1321.44	12.12	1323.61	12.30	1417.24	20.25	1455.55	23.50
75_CLUS_3_2	1138.91	1255.64	10.25	1288.53	13.14	1398.08	22.76	1436.75	26.15
75_CLUS_3_3	1173.96	1312.10	11.77	1324.54	12.83	1427.25	21.58	1474.84	25.63
75_CPCD_3_1	1815.60	2058.79	13.39	2114.13	16.44	2173.03	19.69	2253.30	24.11
75_CPCD_3_2	1810.81	2094.31	15.66	2070.63	14.35	2196.79	21.32	2195.44	21.24
75_CPCD_3_3	1883.97	2152.68	14.26	2158.56	14.58	2224.86	18.09	2257.27	19.81
100_RAND_2_1	3086.28	3523.49	14.17	3530.66	14.40	4020.81	30.28	4070.19	31.88
100_CLUS_2_1	3176.76	3683.07	15.94	3687.00	16.06	4027.97	26.80	4193.50	32.01
100_CPCD_2_1	3541.89	4174.77	17.87	4212.94	18.95	4331.93	22.31	4320.29	21.98
100_RAND_3_1	3110.14	3536.58	13.71	3535.65	13.68	3976.68	27.86	4040.47	29.91
100_CLUS_3_1	3206.88	3686.48	14.96	3714.63	15.83	3986.07	24.30	4155.06	29.57
100_CPCD_3_1	3579.25	4250.80	18.76	4142.83	15.75	4256.80	18.93	4258.44	18.98
Average gap			**11.31**		**11.66**		**20.94**		**25.25**

Table 11: Results (reloading quantities) for different variants of 3L-PDP

Instance	Cargo weight	Variant 1A reloading quantity absolute	Variant 1A reloading quantity in %	Variant 1B reloading quantity absolute	Variant 1B reloading quantity in %	Variant 2 reloading quantity absolute	Variant 2 reloading quantity in %
50_RAND_2_1	610544	575886	94.32	496600	81.34	130697	21.41
50_RAND_2_2	578322	669087	115.69	535546	92.60	133158	23.02
50_RAND_2_3	530415	662901	124.98	612078	115.40	129989	24.51
50_RAND_2_4	652932	566122	86.70	519652	79.59	59762	9.15
50_RAND_2_5	698040	569765	81.62	523570	75.01	146195	20.94
50_CLUS_2_1	610544	520392	85.23	507746	83.16	90256	14.78
50_CLUS_2_2	578322	565259	97.74	527001	91.13	234478	40.54
50_CLUS_2_3	530415	638657	120.41	530535	100.02	130968	24.69
50_CLUS_2_4	652932	630986	96.64	576149	88.24	138649	21.23
50_CLUS_2_5	698040	431940	61.88	422388	60.51	94053	13.47
50_CPCD_2_1	610544	545360	89.32	436470	71.49	41543	6.80
50_CPCD_2_2	578322	590201	102.05	465529	80.50	42577	7.36
50_CPCD_2_3	530415	480357	90.56	416155	78.46	44567	8.40
50_CPCD_2_4	652932	421828	64.61	419679	64.28	42886	6.57
50_CPCD_2_5	698040	413780	59.28	364143	52.17	26928	3.86
50_RAND_3_1	611295	495706	81.09	458559	75.01	89826	14.69
50_RAND_3_2	579037	578519	99.91	509699	88.03	152272	26.30
50_RAND_3_3	531236	681992	128.38	566385	106.62	156420	29.44
50_RAND_3_4	654049	558242	85.35	538405	82.32	40394	6.18
50_RAND_3_5	699080	515608	73.76	487154	69.68	37135	5.31
50_CLUS_3_1	611295	536957	87.84	495350	81.03	156768	25.65
50_CLUS_3_2	579037	588426	101.62	529809	91.50	105480	18.22
50_CLUS_3_3	531236	571454	107.57	565991	106.54	114692	21.59
50_CLUS_3_4	654049	597921	91.42	550284	84.14	110039	16.82
50_CLUS_3_5	699080	574424	82.17	484857	69.36	67788	9.70
50_CPCD_3_1	611295	548726	89.76	452438	74.01	27095	4.43
50_CPCD_3_2	579037	530539	91.62	489940	84.61	46584	8.05
50_CPCD_3_3	531236	486764	91.63	403069	75.87	57034	10.74
50_CPCD_3_4	654049	428243	65.48	367992	56.26	57190	8.74
50_CPCD_3_5	699080	419695	60.04	388517	55.58	55635	7.96
75_RAND_2_1	772435	953854	123.49	764635	98.99	117721	15.24
75_RAND_2_2	780361	907165	116.25	846517	108.48	146900	18.82
75_RAND_2_3	808203	911155	112.74	851973	105.42	160699	19.88
75_CLUS_2_1	772435	836792	108.33	793409	102.72	148422	19.21
75_CLUS_2_2	780361	842717	107.99	715138	91.64	100917	12.93
75_CLUS_2_3	808203	856501	105.98	903500	111.79	163504	20.23
75_CPCD_2_1	772435	843069	109.14	773819	100.18	90743	11.75
75_CPCD_2_2	780361	829112	106.25	657545	84.26	133752	17.14
75_CPCD_2_3	808203	772778	95.62	760018	94.04	68174	8.44
75_RAND_3_1	774140	875228	113.06	772509	99.79	128220	16.56
75_RAND_3_2	782381	983657	125.73	808558	103.35	196943	25.17
75_RAND_3_3	810106	1005567	124.13	874698	107.97	160055	19.76
75_CLUS_3_1	774140	983038	126.98	845074	109.16	146674	18.95
75_CLUS_3_2	782381	874307	111.75	793207	101.38	229418	29.32
75_CLUS_3_3	810106	930583	114.87	887598	109.57	115878	14.30
75_CPCD_3_1	774140	916437	118.38	754219	97.43	132773	17.15
75_CPCD_3_2	782381	842012	107.62	735611	94.02	157132	20.08
75_CPCD_3_3	810106	822820	101.57	699974	86.41	40983	5.06
100_RAND_2_1	1072407	1408169	131.31	1319035	123.00	136278	12.71
100_CLUS_2_1	1072407	1316780	122.79	1124825	104.89	157942	14.73
100_CPCD_2_1	1072407	1132356	105.59	980534	91.43	151785	14.15
100_RAND_3_1	1074809	1404231	130.65	1215821	113.12	161874	15.06
100_CLUS_3_1	1074809	1407984	131.00	1183947	110.15	181252	16.86
100_CPCD_3_1	1074809	1145181	106.55	936335	87.12	107328	9.99
Average			**101.23**		**89.83**		**15.82**

Table 12: Tradeoff between total travel distance and reloading quantity

3L-PDP variant	Total travel distance average in %	Reloading quantity average in %
1A	111.31	101.23
1B	111.66	89.83
2	120.94	15.82
5	125.25	0.00

6 Conclusions and future work

In this paper, the vehicle routing problem with pickup and delivery (PDP) has been extended to an integrated vehicle routing and loading problem with 3D rectangular items to be transported in homogeneous vehicles with a rectangular 3D loading space (3L-PDP). In the problem formulation, we focused on the question under which conditions any reloading effort, i.e. any movement of boxes *after* loading and *before* unloading, can be avoided. It turned out that the RS constraint for loading and unloading sites is not sufficient. Instead, we require either a new packing constraint, termed reloading ban, or a new routing constraint, called independent partial routes condition, to exclude any reloading effort. Eventually, a spectrum of five 3L-PDP variants was introduced that allow for different portions of reloading effort and reciprocal savings of travel distance.

A hybrid algorithm was then proposed to tackle three of the five 3L-PDP variants. It is composed of the adapted large neighborhood search algorithm by Ropke and Pisinger (2006) for the 1D-PDP and the tree search algorithm for packing boxes by Bortfeldt (2012). We tested the algorithm with the 1D-PDPTW instances with 50 and 100 requests by Li and Lim (2001) and found that it is on a par with the best PDP(TW) solution methods available. For testing the hybrid 3L-PDP algorithm 54 3L-PDP instances with up to 100 requests and up to 300 boxes were introduced. The results for the three 3L-PDP variants are plausible in that there is a clear tradeoff between travel distance and reloading effort.

In our future research, we will deal with the residual 3L-PDP variants defined here and including the reloading ban, a packing constraint to preclude any reloading effort. In these problem variants, we can expect even better travel distances given the same assumptions in terms of reloading effort.

References

Baldacci, R; Bartolini, E; Mingozzi, A (2011): An Exact Algorithm for the Pickup and Delivery Problem with Time Windows. *Operations Research*, 59(2):414-426.

Bartók, T; Imre, C (2011): Pickup and Delivery Vehicle Routing with Multidimensional Loading Constraints. *Acta Cybernetica*, 20, 17-33.

Bent, R; van Hentenryck, P (2006): A two-stage hybrid algorithm for pickup and delivery vehicle routing problems with time windows. *Computers & Operations Research*, 33:875-893.

Berbeglia, G; Cordeau, JF; Gribkovskaia, I; Laporte, G (2007): Static pickup and delivery problems: A classification scheme and survey. *Top*, 15:1-31.

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; Männel, D; Mönch, L (2015): Hybrid algorithms for the vehicle routing problem with clustered backhauls and 3D loading constraints. *European Journal of Operational Research*, 243:82-96.

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

Derigs, U; Döhmer, T (2008): Indirect search for the vehicle routing problem with pickup and delivery and time windows. *OR Spectrum*, 30:149-165.

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.

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.

Gendreau, M; Potvin, J Y (2010): Handbook of Metaheuristics, second edition. Springer, New York etc.

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

Iori, M; Martello, S (2013): An Annotated Bibliography of Combined Routing and Loading Problems. *Yugoslav Journal of Operations Research*, 23(3):311-326.

Koning, D (2011): Using Column Generation for the Pickup and Delivery Problem with Disturbances. *Technical Report, Department of Computer Science, Utrecht University*.

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.

Li, H; Lim, A (2001): A metaheuristic for the pickup and delivery problem with time windows. 13th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'01). IEEE Computer Society, Los Alamitos, CA, 333-340.

Lim, H; Lim, A; Rodrigues, B (2002): Solving the pickup and delivery problem with time windows using squeaky wheel optimization with local search. American Conference on Information Systems, AMICS 2002, Dallas, USA.

Lu, Q; Dessouky, MM (2006): A new insertion-based construction heuristic for solving the pickup and delivery problem with time windows. *European Journal of Operational Research*, 175:672-687.

Malapert, A; Guéret, C; Jussien, N; Langevin, A; Rousseau, LM (2008): Two-dimensional Pickup and Delivery Routing Problem with Loading Constraints. *Proceedings of the First CPAIOR Workshop on Bin Packing and Placement Constraints (BPPC'08)*, Paris, France.

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

Nagata, Y; Kobayashi, S (2010): A Memetic Algorithm for the Pickup and Delivery Problem with Time Windows Using Selective Route Exchange Crossover. *Parallel Problem Solving from Nature, PPSN XI*, R. Schaefer et al. (eds.), 536-545, Springer: Berlin.

Nanry, WP; Barnes, W (2000): Solving the pickup and delivery problem with time windows using reactive tabu search. *Transportation Research Part B: Methodological*, 34:107-121.

Pankratz, G (2005): A grouping algorithm for the pickup and delivery problem with time windows. *OR Spectrum*, 27:21-41.

Parragh, SN; Doerner, KF; Hartl, RF (2008): A Survey on Pickup and Delivery Problems. Part II: Transportation between pickup and delivery locations. *Journal für Betriebswirtschaft*, 58:81-117.

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

Ropke, S; Cordeau, JF (2009): Branch and Cut and Price for the Pickup and Delivery Problem with Time Windows. *Transportation Science*, 43(3):267-286.

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

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.

Sintef (2015): www.sintef.no/projectweb/top/pdptw/li--lim-benchmark, last visited 01.04.2015

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

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.

Toth, P; Vigo, D (2014): Vehicle Routing: Problems, Methods, and Applications, second edition. MOS-SIAM series on optimization, Philadelphia.

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.

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.

Xu, H; Chen, ZL; Rajagopal, S; Arunapuram, S (2003): Solving a practical pickup and delivery problem. *Transportation Science*, 37:347-364.

Zachariadis, E; Tarantilis, C; Kiranoudis, C (2012): The pallet-packing vehicle routing problem. *Transportation Science*, 46 (3): 341-358.

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.

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

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

www.fww.ovgu.de/femm

ISSN 1615-4274