ECONSTOR
Make Your Publications Visible.

A Service of
ZBW
Leibniz-Informationszentrum Wirtschaft
Leibniz Information Centre for Economics

Krebs, Corinna; Ehmke, Jan Fabian; Koch, Henriette

Article — Published Version

Advanced loading constraints for 3D vehicle routing problems

OR Spectrum

Provided in Cooperation with:
Springer Nature

Suggested Citation: Krebs, Corinna; Ehmke, Jan Fabian; Koch, Henriette (2021) : Advanced loading constraints for 3D vehicle routing problems, OR Spectrum, ISSN 1436-6304, Springer, Berlin, Heidelberg, Vol. 43, Iss. 4, pp. 835-875, https://doi.org/10.1007/s00291-021-00645-w

This Version is available at:
https://hdl.handle.net/10419/286795

Standard-Nutzungsbedingungen:

Die Dokumente auf EconStor dürfen zu eigenen wissenschaftlichen Zwecken und zum Privatgebrauch gespeichert und kopiert werden.

Sie dürfen die Dokumente nicht für öffentliche oder kommerzielle Zwecke vervielfältigen, öffentlich ausstellen, öffentlich zugänglich machen, vertreiben oder anderweitig nutzen.

Sofern die Verfasser die Dokumente unter Open-Content-Lizenzen (insbesondere CC-Lizenzen) zur Verfügung gestellt haben sollten, gelten abweichend von diesen Nutzungsbedingungen die in der dort genannten Lizenz gewährten Nutzungsrechte.

Terms of use:

Documents in EconStor may be saved and copied for your personal and scholarly purposes.

You are not to copy documents for public or commercial purposes, to exhibit the documents publicly, to make them publicly available on the internet, or to distribute or otherwise use the documents in public.

If the documents have been made available under an Open Content Licence (especially Creative Commons Licences), you may exercise further usage rights as specified in the indicated licence.

https://creativecommons.org/licenses/by/4.0/

WWW.ECONSTOR.EU

Mitglied der Leibniz-Gemeinschaft

OR Spectrum (2021) 43:835–875
https://doi.org/10.1007/s00291-021-00645-w

ORIGINAL ARTICLE

Advanced loading constraints for 3D vehicle routing problems

Corinna Krebs[1] · **Jan Fabian Ehmke**[2] · **Henriette Koch**[1]

Received: 24 June 2020 / Accepted: 5 July 2021 / Published online: 11 August 2021
© The Author(s) 2021

Abstract

Given automated order systems, detailed characteristics of items and vehicles enable the detailed planning of deliveries including more efficient and safer loading of distribution vehicles. Many vehicle routing approaches ignore complex loading constraints. This paper focuses on the comprehensive evaluation of loading constraints in the context of combined Capacitated Vehicle Routing Problem and 3D Loading (3L-CVRP) and its extension with time windows (3L-VRPTW). To the best of our knowledge, this paper considers the currently largest number of loading constraints meeting real-world requirements and reducing unnecessary loading efforts for both problem variants. We introduce an approach for the load bearing strength of items ensuring a realistic load distribution between items. Moreover, we provide a new variant for the robust stability constraint enabling better performance and higher stability. In addition, we consider axle weights of vehicles to prevent overloaded axles for the first time for the 3L-VRPTW. Additionally, the reachability of items, balanced loading and manual unloading of items are taken into account. All loading constraints are implemented in a deepest-bottom-left-fill algorithm, which is embedded in an outer adaptive large neighbourhood search tackling the Vehicle Routing Problem. A new set of 600 instances is created, published and used to evaluate all loading constraints in terms of solution quality and performance. The efficiency of the hybrid algorithm is evaluated by three well-known instance sets. We outperform the benchmarks for most instance sets from the literature. Detailed results and the implementation of loading constraints are published online.

Keywords 3L-CVRP · 3L-VRPTW · Loading constraints · Load bearing strength · Stability

✉ Corinna Krebs
Corinna.Krebs@ovgu.de

Extended author information available on the last page of the article

1 Introduction

In recent years, sales in online trading have risen steadily. Forecasts for the coming years predict significant growth. Therefore, efficient logistics operations are more important than ever. Through many years of research in the field of Vehicle Routing Problems (VRP), (near-) optimal tour plans can be found for many use cases. Hereby, the demand of a customer is often simplified by using a total mass or volume for the items to be delivered. In practice, solutions might be infeasible since a vehicle cannot be feasibly packed because of unbalanced loading and/or unsafe placement of items. As more and more information on items becomes available for detailed planning, the realistic planning of transportation and of packing processes could become the key factor for cost reduction and safety, leading to an increasing interest in combined routing and loading problems.

The combined problem at hand is the Three-Dimensional Loading Capacitated Vehicle Routing Problem (3L-CVRP). It was first introduced by Gendreau et al. (2006) and assumes delivery of cuboid items laying at the depot. A homogeneous fleet of vehicles is available for transporting the items to a number of customers. Each vehicle must be equipped with a feasible packing plan considering several loading constraints. The depot and the customers have specific time windows, in which the delivery must take place. This problem variant is known as the Three-Dimensional Loading Vehicle Routing Problem with Time Windows (3L-VRPTW).

The focus of this paper is on the comprehensive examination of loading constraints. Although the consideration of different complex loading constraints leads to more realistic models, it is mainly neglected in work dealing with the 3L-CVRP problem or its variants so far. The reason is that modelling and evaluation of loading constraints are complex and require new solution approaches. We tackle this problem and integrate the current largest constraint set so far. We introduce a new variant for the robust stability constraint, which increases the stability and the performance. Moreover, we develop an approach based on the science of statics so that for the first time, the acting load on an item is distributed through the entire stack, which ensures realistic and stable packing plans. In addition, this paper considers aspects for manual unloading, reachability, the axle weights of vehicles and a balanced loading. For the latter, we introduce formulas to illustrate our implementation approach. In case of manual unloading, the items are unloaded without lifting. The reachability constraint avoids unnecessary rearrangements of items. Detailed modelling of axle weights and balanced loading prevents overloaded axles and tipping over of vehicles. The implementation of all loading constraints is published online within a solution validator written in C++ as well as in Java. The validator can be used to check the feasibility of solutions for different loading constraint sets.

All constraints are integrated in a hybrid algorithm. The hybrid algorithm consists of an inner deepest-bottom-left-fill algorithm which solves the Loading Problem and is embedded in an outer adaptive large neighbourhood search tackling the Vehicle Routing Problem. The efficiency of the hybrid algorithm is

shown by using the instance sets by Ceschia et al. (2013), Moura and Oliveira (2009) and Zhang et al. (2017). Experiments show that the hybrid algorithm performs better than the benchmark for most instance sets.

Moreover, we have created and published an instance set consisting of 600 new instances varying systematically in the number of customers, of item types and of items. For the first time, every complex loading constraint is evaluated concerning its impact on the objective values (number of used vehicles and total travel distance) grouped by number of item types, items and customers. Our evaluations consist of over 30,000 results, and we provide all results online and in detail (e.g. routing and packing plans with the position of all items) to ensure extraordinary transparency. On this basis, we give recommendations about which constraints are reasonable based on their impact on algorithmic performance and solution quality.

The paper is organized as follows. In Sect. 2, the relevant literature is reviewed. The 3L-CVRP and the 3L-VRPTW are formulated in Sect. 3. In Sect. 4, the new definitions and the implementation variants of the loading constraints are presented. In Sect. 5, the hybrid algorithm is described, and Sect. 6 deals with the testing of the constraints. Finally, conclusions are drawn in Sect. 7.

2 Literature review

This paper considers the Three-Dimensional Loading Capacitated Vehicle Routing Problem (3L-CVRP) and its extension with Time Windows (3L-VRPTW), which represent a combination of the Vehicle Routing Problem (VRP) and 3D Loading constraints. As shown in Table 2, the consideration of multiple loading constraints is currently sparely researched. Thus, this paper examines the impact of different loading constraints on the results for the 3L-CVRP and the 3L-VRPTW. The current state of modelling loading constraints for both problems is analysed in the following.

2.1 3L-CVRP

Gendreau et al. (2006) introduced the combined Vehicle Routing and 3D Loading Problem, namely 3L-CVRP. They solve the VRP using an ”outer” tabu search, which determines customer sequences. An iteratively invoked ”inner” tabu search defines the item sequence for the routes. The loading algorithms are based on the touching parameter algorithm by Lodi et al. (1999) and the bottom-left-algorithm by Baker et al. (1980). The items are packed orthogonally into the vehicle loading space (orthogonality constraint) without overlapping and respecting their dimensions (geometry constraint). The rotation of the items is only allowed along the width-length plane (rotation constraint). Each item has a mass, and the vehicle has a maximum capacity (load capacity). Moreover, a fragility flag is assigned to each item to prevent stacking fragile items on top of each other (fragility constraint). When stacking items, they must be supported by other items with a certain percentage (minimal supporting area constraint). When unloading items, it should be done by direct movements parallel to the length of the vehicle (LIFO constraint). Since

the constraints orthogonality, geometry, rotation, load capacity, fragility, minimal supporting area and LIFO are commonly considered in researches on the 3L-CVRP and its variants, this set is here defined as *basic constraint set*. For testing, Gendreau et al. (2006) developed 27 instances.

The 3L-CVRP has been studied intensively in recent years so that the results for this benchmark have been improved repeatedly (e.g. Tarantilis et al. 2009; Fuellerer et al. 2010; Bortfeldt 2012 and Wei et al. (2014)). Tarantilis et al. (2009) used a combination of tabu search and guided local search to build the routes. For the Loading Problem, successively six packing heuristics are called until a feasible solution is found. They also present a new variant—the Capacitated Vehicle Routing Problem with Manual 3D Loading Constraints (M3L-CVRP). This variant deals with the manual handling of items, e.g. the items are small and of low mass. Therefore, the LIFO constraint is modified so that it is allowed that one item hangs over another one. This adaption of the LIFO policy, which is in this paper referred to as MLIFO, is also examined in a paper by Ceschia et al. (2013). Ceschia et al. propose a local search approach combining simulated annealing and large neighbourhood search to solve the VRP. To handle the Loading Problem, one out of nine loading heuristics based on the bottom-left-algorithm and the touching perimeter algorithm is selected. Besides the MLIFO constraint, they consider the reachability of an item for the first time within the 3L-CVRP. In the context of the Three-Dimensional Bin Packing Problem, this constraint was developed by Junqueira et al. (2013) to avoid the driver standing on items to reach other items for unloading or arranging operations. Ceschia et al. (2013) also include the item's load bearing strength (lbs), which was first mentioned by Bischoff and Ratcliff (1995) and examined in Bischoff (2003) for the Three-Dimensional Bin Packing Problem. Thus, to the best of our knowledge, Ceschia et al. currently combine the most loading constraints. Krebs and Ehmke (2021) consider detailed modelling of axle weights of vehicles for the first time for the 3L-CVRP.

2.2 3L-VRPTW

In Moura (2008) and Moura and Oliveira (2009), the VRTWLP is introduced, which corresponds to the 3L-VRPTW without the consideration of masses and stacking constraints (e.g. fragility and load capacity) and with higher stability requirements (full support) and with more rotation possibilities. Moura (2008) proposes a multi-objective genetic algorithm to generate routes (VRP). If a customer is inserted in a route, a wall-building heuristic is called to tackle the Loading Problem. This packing heuristic is also used in Moura and Oliveira (2009), where a hierarchical and a sequential approach are combined. The hierarchical one solves primarily the VRP, while the sequential one handles the VRPTW and the bin packing. 46 instances are created. The current best-known results for these instances are received by Reil et al. (2018), who solve the packing problem through a tabu search algorithm. Then, a multi-start evolutionary search minimizes the number of used vehicles while another tabu search algorithm minimizes the total travel distance. Pace et al. (2015) propose a heuristic based on simulated annealing and an iterated local search for the routing

phase. Since they examine the distribution of fibre boards, a specialized loading heuristic based on a depth-first tree search and a balanced loading constraint are necessary. The latter is also adopted by Mak-Hau et al. (2018), who develop a mixed-integer linear programme model of the 3L-VRPTW with a heterogeneous fleet. Zhang et al. (2017) solve the 3L-VRPTW with a hybrid approach, consisting of a new loading heuristic and a routing heuristic based on a tabu search and an artificial bee colony algorithm. They include the *basic constraint set* and combine the two well-known instance sets provided by Gendreau et al. (2006) and Solomon (1987).

In this paper, we use the approach by Koch et al. (2018) proposed for the 3L-VRPTW with Backhauls, which is also used for the 3L-CVRP in Krebs and Ehmke (2021). The following Table 1 summarizes the approaches.

Table 2 summarizes the related literature and highlights our contribution. As demonstrated in Table 2, this paper deals with the largest constraints set and combines the robust stability (C6b), load bearing strength (C7b) and reachability (C8) with axle weights (C9) and the balanced loading (C10) constraints. Moreover, we distribute the loads for the first time through the entire stack in the load bearing strength constraint.

3 Problem formulation

Following the convention by Koch et al. (2018), the 3L-VRPTW is described as follows: Let $G = (N, E)$ be a complete, directed graph, where N is the set of $n+1$ nodes including the depot (node 0) and n customers to be served (node 1 to n), and E is the edge set connecting each pair of nodes. Each edge $e_{i,j} \in E$ $(i \neq j, i, j = 0, ..., n)$ has an associated routing distance $d_{i,j}$ $(d_{i,j} > 0)$. The demand of customer $i \in N \setminus \{0\}$ consists of c_i cuboid items. Let m be the total number of all demanded items. Moreover, time windows are considered by assigning three times to each node i: the ready time RT_i, which is the earliest possible start time of service, the due date DD_i, the latest possible start time, and the service time ST_i, which specifies the needed time to (un-) load all c_i items of a customer i.

Each item $I_{i,k}$ $(k = 1, ..., c_i)$ is defined by mass $m_{i,k}$, length $l_{i,k}$, width $w_{i,k}$ and height $h_{i,k}$. The items are delivered by at most v_{max} available, homogenous vehicles. Each vehicle has a maximum load capacity D and a cuboid loading space defined by length L, width W and height H. It is assumed that each vehicle has a constant speed of 1 distance unit per time unit. If a vehicle arrives at an edge before its ready time, it has to wait until the ready time is reached.

Let v_{used} be the number of used vehicles in a solution. A solution is a set of v_{used} pairs of routes R_v and packing plans PP_v, whereby the route R_v $(v = 1, ..., v_{used})$ is an ordered sequence of at least one customer and PP_v is a packing plan containing the position within the loading space for each item included in the route.

A solution is feasible if

(S1) All routes R_v and packing plans PP_v are feasible (see below);
(S2) Each customer is visited exactly once;

Table 1 Summary and overview of approaches

Author	Problem	Routing approach	Loading approach	Stopping criterion	New instances
Gendreau et al. (2006)	3L-CVRP	Tabu search	Touching perimeter algorithm, bottom-left algorithm	Time limit	27
Tarantilis et al. (2009)	3L-CVRP	Tabu search, guided local search	Six heuristics	Iterations without improvement	12
Fuellerer et al. (2010)	3L-CVRP	Savings-based ACO algorithm	Touching perimeter algorithm, bottom-left algorithm	Maximum iterations	
Bortfeldt (2012)	3L-CVRP	Tabu search	Tree search	Time limit, maximum iterations	
Wei et al. (2014)	3L-CVRP, 3L-HFVRP	Adaptive variable neighbourhood search	Extreme point based first fit	Time limit	36
Ceschia et al. (2013)	3L-CVRP	Large-neighbourhood search, simulated annealing	Nine heuristics based on touching perimeter algorithm, bottom-left algorithm, wall-building heuristic	Time limit, maximum iterations	13
Moura (2008)	VRTWLP	Multi-objective genetic algorithm	Wall-building heuristic	Iterations without improvement	
Moura and Oliveira (2009)	VRTWLP	Heuristic with hierarchical and sequential approaches	Wall-building heuristic	Iterations without improvement	46
Pace et al. (2015)	3L-VRPTW	Iterated local Search, simulated annealing	Depth-first tree search	Maximum iterations	
Zhang et al. (2017)	3L-VRPTW	Tabu search, artificial bee colony algorithm	New loading heuristic	Maximum iterations	27
Mak-Hau et al. (2018)	3L-VRPTW, Split Delivery	Mixed-integer linear programm model		Time limit	
Reil et al. (2018)	VRTWLP	Multi-start evolutionary search, tabu search	Tabu search	Time limit, iterations without improvement	
Krebs and Ehmke (2021)	3L-CVRP	ALNS	DBLF	Time limit, iterations without improvement, maximum iterations	

Table 2 Summary and overview over loading constraints

References sorted by year	Geometry C1	Orthogo-nality C2	Rotation C3	Load capacity C4	LIFO C5a	MLIFO C5b	Minimal supporting area C6a	Robust stability C6b	Fragility C7a	Load bearing strength C7b	Reach-ability C8	Axle weights C9	Balanced loading C10
Gendreau et al. (2006)	✓	✓	✓	✓	✓		✓		✓				
Moura (2008)	✓	✓	✓		✓		✓						
Moura and Oliveira (2009)	✓	✓	✓		✓		✓						
Tarantilis et al. (2009)	✓	✓	✓	✓	✓	✓	✓		✓				
Fuellerer et al. (2009)	✓	✓	✓	✓	✓		✓		✓				
Bortfeldt (2012)	✓	✓	✓	✓	✓		✓		✓				
Ceschia et al. (2013)	✓	✓	✓	✓	✓	✓	✓	✓	✓	(✓)	✓		
Pace et al. (2015)	✓	✓		✓	✓		✓		✓				✓
Zhang et al. (2017)	✓	✓	✓	✓	✓		✓		✓				
Mak-Hau et al. (2018)	✓	✓		✓	✓								✓
Koch et al. (2018)	✓	✓	✓	✓	✓		✓		✓				
Reil et al. (2018)	✓	✓	✓	✓	✓		✓		✓				
Krebs and Ehmke (2021)	✓	✓	✓	✓	✓		✓		✓			✓	
This paper	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

(S3) The number of used vehicles v_{used} does not exceed the number of available vehicles v_{max};
(S4) Each packing plan PP_v contains all c_i items of all customers i included in the corresponding route ($i \in R_v$).

A route R_v must meet the following routing constraints:

(R1) Each route starts and terminates at the depot and visits at least one customer;
(R2) The vehicle does not arrive after the due date DD_i of any location i.

Each packing plan must obey a loading set P defining a subset of the following loading constraints, which are described in detail in next Sect. 4.

(C1) *Geometry*: The items must be packed within the vehicle without overlapping;
(C2) *Orthogonality*: The items can only be placed orthogonally inside a vehicle;
(C3) *Rotation*: The items can be rotated 90° only on the width-length plane;
(C4) *Load capacity*: The sum of masses of all included items of a vehicle does not exceed the maximum load capacity D.
(C5a) *LIFO*: No item is placed above or in front of item $I_{i,k}$, which belongs to a customer served after customer i;
(C5b) *MLIFO*: No item is placed on or in front of item $I_{i,k}$, which belongs to a customer served after customer i;
(C6a) *Minimal supporting area*: Each item has a supporting area of at least a percentage α of its base area;
(C6b) *Robust stability*: Each item has a supporting area of at least a percentage α of its base area at any height;
(C7a) *Fragility*: No non-fragile items are placed on top of fragile items;
(C7b) *Load bearing strength*: The load bearing strength $lbs_{i,k}$ is the maximal load per area unit an item can bear. It must not be exceeded anywhere on the top face of an item;
(C8) *Reachability*: The distance between an item and the driver must be less or equal than a certain length λ;
(C9) *Axle weights*: The loads for the front and the rear axle do not exceed the permissible axle weights FA_{perm} and RA_{perm};
(C10) *Balanced loading*: The load of one vehicle half does not exceed a certain percentage p of D.

The 3L-CVRP and 3L-VRPTW aim at determining a feasible solution minimizing the objective values, e.g. number of used vehicles v_{used} and the total travel distance ttd, and meeting all corresponding constraints.

4 Definitions and implementations of loading constraints

This section discusses the implementation details and challenges of the considered loading constraints. We introduce new realizations and implementation variants. Detailed algorithms are provided and explained in a solution validator, written in Java and C++, available via http://github.com/CorinnaKrebs/SolutionValidator.

Table 3 gives an overview of the considered loading constraints. The loading constraints C1-C4, C5a, C6a and C7a are used as described in Gendreau et al. (2006). In Table 3, we have highlighted new developed loading constraints in bold and constraints examined for the first time for the 3L-VRPTW in italics.

4.1 Unloading sequence (C5)

The unloading sequence constraints define the order in which the items of the customers of one route should be unloaded. The purpose is to prevent costly reloading processes of items during the unloading process. In the following, two definitions, namely LIFO (C5a) and MLIFO (C5b), are shown.

4.1.1 LIFO (C5a)

As shown in Gendreau et al. (2006), the last-in first-out (LIFO) constraint treats the unloading sequence in the way that all items c_i of a customer i are loaded and unloaded by movements parallel to the front-rear axis (x-axis) of the vehicle without moving other items. Forklifts are mostly used for this purpose, which may need to lift an item during the unloading process (cf. Ceschia et al. 2013). Therefore, no item

Table 3 Overview of loading constraints

Abbr.	Constraint	Definition	Variant
C1	Geometry		
C2	Orthogonality		
C3	Rotation		
C4	Load capacity		
C5a	Unloading sequence	LIFO	
C5b	Unloading sequence	MLIFO	
C6a	Vertical stability	Minimal supporting area	
C6b1	Vertical stability	Robust stability	*Multiple overhanging*
C6b2	Vertical stability	Robust stability	**Top overhanging**
C7a	Stacking	Fragility	
C7b1	Stacking	Load bearing strength	*Simplified selection*
C7b2	Stacking	Load bearing strength	**Complete selection**
C8	Reachability		
C9	*Axle weights*		
C10	*Balanced loading*		

demanded by a customer that is delivered later can be placed over $I_{i,k}$ or between $I_{i,k}$ and the rear of the vehicle.

4.1.2 MLIFO (C5b)

In the Manual LIFO constraint (MLIFO) introduced by Tarantilis et al. (2009), the items are (un-)loaded by manual operations without the usage of, e.g. forklifts. Consequently, the items can be (un-)loaded without lifting them. Therefore, an item demanded by a customer that is served later than customer i can hang over the item $I_{i,k}$ without touching its surface and without being placed between $I_{i,k}$ and the rear of the vehicle.

The differences between the LIFO and the MLIFO constraint are visualized in Fig. 1. For both variants, it is not allowed to place an item directly on top of another item that is delivered earlier (see Fig. 1a). In contrast to the LIFO constraint, it is allowed that one item hangs over another item that is delivered earlier (see Fig. 1b).

4.2 Vertical stability (C6)

The vertical stability constraints prevent stacked items from falling on the ground. For this purpose, we show that the current definition is not sufficient and formulate the new robust stability constraint.

4.2.1 Minimal supporting area (C6a)

The minimal supporting area constraint ensures that a certain ratio α of the base of a stacked item is supported by the upper surface of the directly underlying items (see Gendreau et al. 2006). As shown in Fig. 2, this formulation can lead to unstable, but still feasible item arrangements: When stacking several items with same

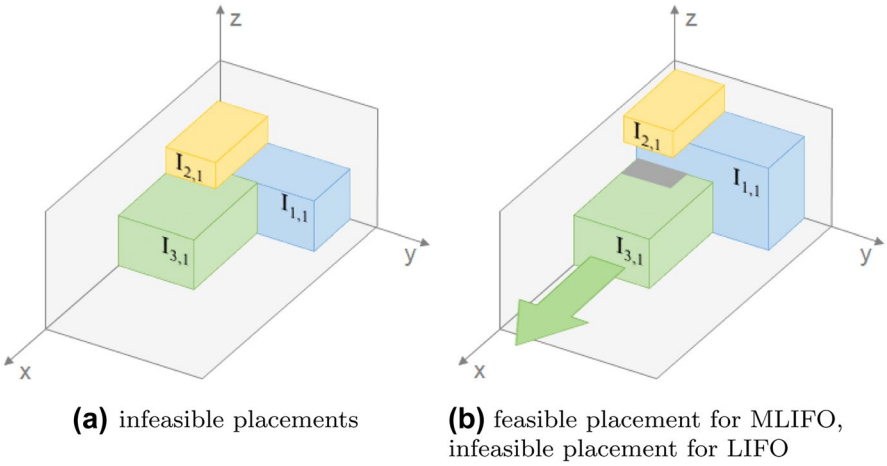


Fig. 1 Difference between LIFO and MLIFO

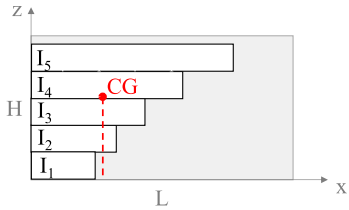


Fig. 2 Unstable, feasible stack w.r.t. minimal supporting area

density, whereby the length or width of each item enlarge by $\frac{1}{\alpha}$, an overhanging stack of items is created.

This arrangement is in accordance with the minimal supporting area constraint (C6a) because for the calculation of the support for one item, only the directly underlying items are considered. According to the science of statics, this stack is not stable, because the x-value of the centre of gravity (CG) lays outside of the dimensions of the first item. Therefore, the stack would topple.

4.2.2 Robust stability (C6b)

As shown above, the minimal supporting area can lead to unstable stacks, since in the calculation of the item's support only the directly underlying items are considered. Therefore, we formulate the robust stability constraint as follows: For each item, the relative support of at least a percentage of α needs to be guaranteed at any height from the vehicle ground to the item's bottom edge.

Multiple overhanging (C6b1): This constraint was first introduced by Ceschia et al. (2013). As the name suggests, all items of a stack are allowed to overhang. When placing an item, the minimal supporting area is checked for all underlying items: Let U be the set which includes all placed items supporting directly or indirectly the item $I_{i,k}$. An item I_u supports $I_{i,k}$ directly if the top area of item I_u has direct contact with the base area of item $I_{i,k}$. An item I_u supports $I_{i,k}$ indirectly if I_u directly supports any placed item which directly supports $I_{i,k}$. Each coordinate for the top surface of item $I_a \in U$ defines a plane. Another item $I_b \in U$ counts to this plane if the top surface of I_b is at the same level as of the plane (see items $I_{1,2}$ and $I_{1,3}$ in Fig. 3b) or if the top surface of I_b is above the plane and the base area of I_b is below the plane (see item $I_{1,3}$ in Fig. 3c). Each plane must obey the minimal supporting

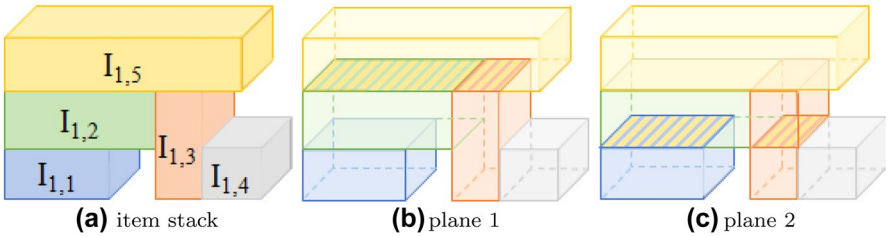


Fig. 3 Determination of planes for item $I_{1,5}$

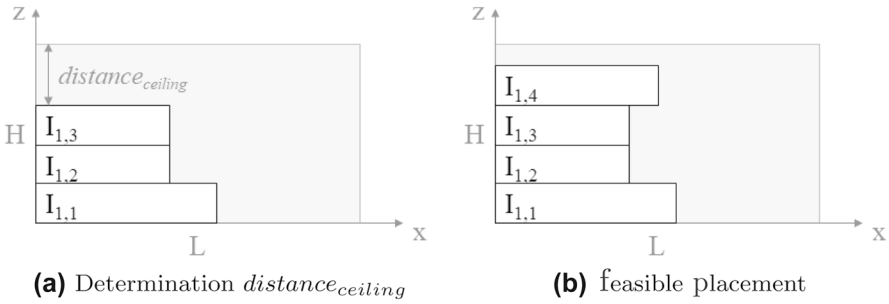


Fig. 4 Implementation of robust stability—top overhanging

area constraint. Otherwise, the constraint is violated and the placement of item $I_{i,k}$ is rejected.

Top overhanging (C6b2): In this paper, we want to introduce another variant for the robust stability, namely the "top overhanging" constraint. In contrast to the previous approach, here, only the topmost item of a stack is allowed to hang over other items. Hence, all items of a stack must be completely supported by other items except the topmost item, which can hang over considering the minimal supporting area constraint (C6a) (see Fig. 4b). This is appropriate for high stability requirements.

Top overhanging is implemented in the following way: Let $distance_{ceiling}$ be the distance between the topmost item of a stack and the ceiling (see Fig. 4a). Let h_{min} be the smallest height of any unplaced item I_{min} of the route and $I_{i,k}$ be the item which should be placed on top of the stack. When stacking items, two cases can occur:

1. If $distance_{ceiling} + h_{i,k} \geq h_{min}$, then item $I_{i,k}$ as well as I_{min} can be placed on the stack. In this case, the item $I_{i,k}$ must be fully supported, since I_{min} could be placed on top of $I_{i,k}$, so that $I_{i,k}$ is not the topmost item of the stack.
2. If $distance_{ceiling} + h_{i,k} < h_{min}$, then no unplaced item can be stacked on top of the stack. Thus, the item $I_{i,k}$ is the topmost item and must therefore obey the minimal supporting area constraint (C6a).

4.3 Stacking (C7)

The stacking constraints focus on the ability of items to bear other items. In the following, different approaches are shown. The fragility constraint (C7a) as shown in Gendreau et al. (2006) is the standard approach. The load bearing strength constraint (C7b) is proposed by Bischoff (2003) for the Container Loading Problem, where each item has an additional parameter indicating the maximum load it can bear. For this load bearing strength constraint, two implementation variants are described below. The first one (C7b1) is proposed by Bischoff (2003), while another

approach is developed and introduced in this paper and is based on the science of statics (C7b2).

4.3.1 Fragility (C7a)

As shown in Gendreau et al. (2006), a fragility flag $f_{i,k}$ is assigned to each item to divide them into fragile items ($f_{i,k} = 1$) and non-fragile ones ($f_{i,k} = 0$). On top of a fragile item, only another fragile item can be stacked, whereas both fragile and non-fragile items can be stacked on a non-fragile item. As demonstrated in Ceschia et al. (2013), the fragility constraint (C7a) has weaknesses: It is supposed that a non-fragile item lies mostly on another non-fragile item and a very small part on a fragile one (see Fig. 5). Even if the non-fragile part on top of the fragile item would be infinitely small, the arrangement remains infeasible.

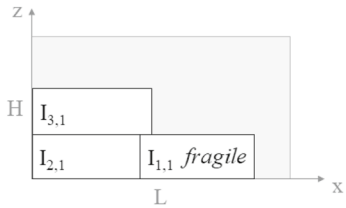


Fig. 5 Infeasible item arrangement w.r.t. fragility constraint

4.3.2 Load bearing strength (C7b)

To handle the issue described before, the actual load on the items should be considered. Therefore, the load bearing strength (LBS) constraint is introduced: Each item $I_{i,k}$ can support a maximum load per area described by the parameter $lbs_{i,k}$. It must not be exceeded anywhere on the top face of an item. A small $lbs_{i,k}$ value corresponds to fragile items.

If an item I_c is stacked on top of another item I_u, then, a load caused by I_c acts on the underlying item I_u ($load_{c,u}$). For its calculation, the percentage of support for item I_c ($support_c$) provided by all directly underlying items must be first determined. Then, all area units ($supportArea_{c,u}$) between Item I_c and I_u must be identified.

Based on that, the support share of I_u on I_c is given as follows:

$$support_{c,u} = \frac{supportArea_{c,u}}{l_c \cdot w_c}. \tag{1}$$

Since the item I_c could overhang, but the load must be distributed in total, the support share is increased proportionally:

$$support_{prop} = \frac{support_{c,u}}{support_c}. \tag{2}$$

The load acting on item I_u is given as follows:

$$load_{c,u} = support_{prop} \cdot load_c, \tag{3}$$

where $load_c$ is the load which has to be distributed due to item I_c. Its value is explained below.

When placing an item on top of another, then the load must be distributed to underlying items. There are two ways to select these items: the simplified and the complete selection.

Simplified selection (C7b1): The approach proposed by Bischoff (2003) selects all items which are underneath the base area (e.g. the footprint) of an item I_c. When placing an item I_c on top of a stack, then all items which are underneath the base area and which directly or indirectly support item I_c are considered. In this case, not all items of the stack may contribute to the mass distribution (see item $I_{1,3}$ in Fig. 6). In this approach, $load_c$ in Eq. 3 corresponds to the mass of I_c. The example in Fig. 6 shows the resulting loads for the underlying items caused only by item $I_{1,6}$.

Complete selection (C7b2): The following approach is based on the science of statics. When placing an item I_c on top of other items, all items are investigated that are located directly below item I_c. Therefore, the mass of I_c is distributed as $load_c$ to the directly underlying items. Then, for each of these items, the received $load_c$ is further adopted and distributed to the directly underlying items again. This is recursively

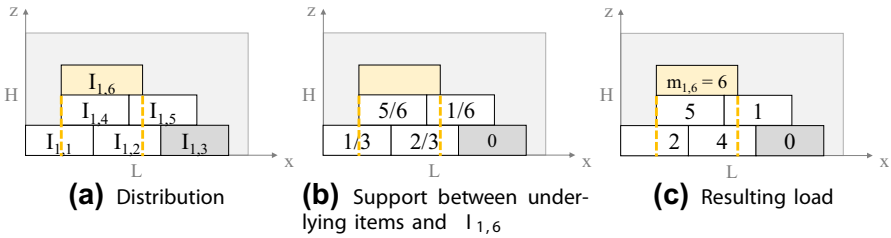


Fig. 6 Mass distribution according to simplified selection based on item $I_{1,6}$

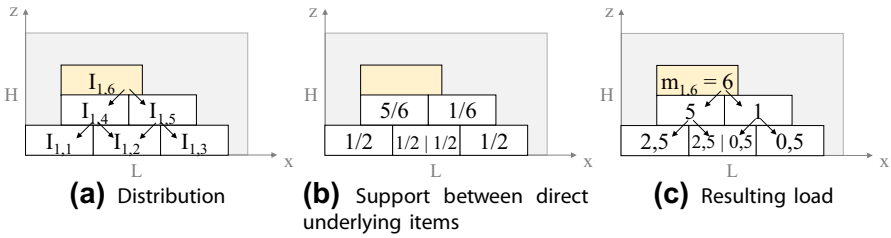


Fig. 7 Mass distribution according to complete selection based on item $I_{1,6}$

repeated until the items on the ground are reached. Fig. 7 shows the same exemplary situation as Fig. 6.

In this approach, all items of a stack contribute to the mass distribution. The resulting loads caused by item $I_{1,4}$ and item $I_{1,5}$ are calculated in the same way.

4.4 Reachability (C8)

When an item is (un-)loaded, then it should be guaranteed that the working equipment or the driver can reach the item when standing as close as possible to the item (cf. Junqueira et al. 2013). For this purpose, the distance $r_{i,k}$ of an item $I_{i,k}$ should be equal or less than a certain length λ, which represent the driver's arm length, for example.

In this paper, for the reachability of an item $I_{i,k}$, all items of customers, which are served after customer i and placed above or beneath item $I_{i,k}$, are considered (see Fig. 8a). The distance $r_{i,k}$ is defined by the front of the item which is the closest to the door (*MaxFront*) and the front of item $I_{i,k}$.

If the distance is larger than λ and thus the item is not reachable, then it is tried to shift the item along the x-axis. This is achieved by searching for the maximum x-value of already placed items on the same layer (*MaxShift*). The new placement must obey the DBL policy. Therefore, the item $I_{i,k}$ is shifted until the reachability constraint is just fulfilled, which means the new distance is defined by $MaxFront - \lambda$ (see Fig. 8b). Additionally, the new placement is tested w.r.t. the loading constraint set P. If the item is not reachable and it cannot be shifted, the placement is rejected.

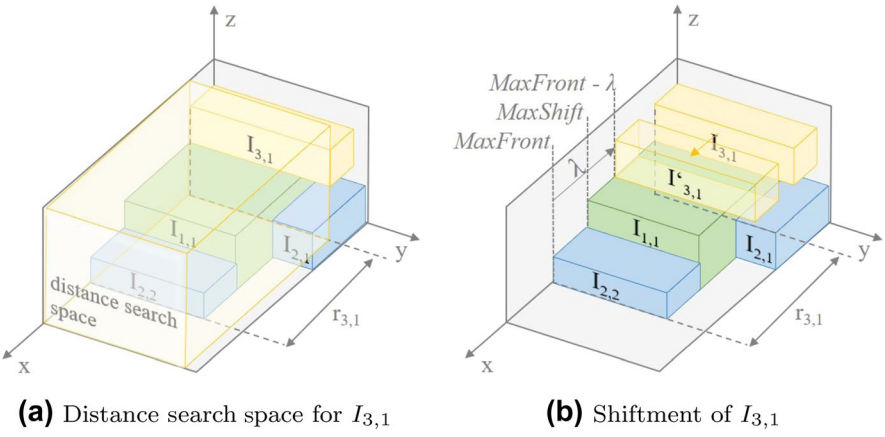


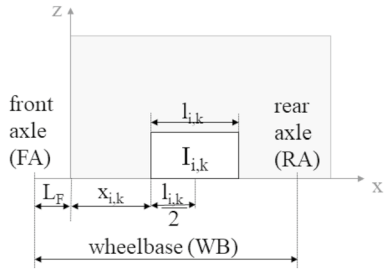
Fig. 8 Illustration of the distance search space for $I_{3,1}$

4.5 Axle weights (C9)

The exceedance of the maximum axle weights of one or more axles leads to far-reaching consequences with regard to vehicle safety: It increases the braking distance

and, in the event of a collision, the consequences are more severe due to the increased impact energy. Therefore, the axle weights constraint respects the permissible axle weights for the front and rear axles of a vehicle. Let FA_{perm} be the maximum load the vehicle's front axle can bear and RA_{perm} be the maximum load for the rear axle, respectively. Both limits are given in mass units. Let L_f be the length between the front axle and the loading space (see Fig. 9). The wheelbase WB is the distance between the front and the rear axle. For each placed item $I_{i,k}$ at the x-position $x_{i,k}$, the distance $s_{i,k}$ between the mass centre of $I_{i,k}$ and the front axle must be determined.

Fig. 9 Vehicle data



According to the approach by Krebs and Ehmke (2021), the following formulas can be applied for a vehicle v to calculate the acting forces for the front F_{FA} and the rear F_{RA} axle. Hereby, g is the constant for acceleration of gravity ($g \approx 9.81 \frac{m}{s^2}$).

$$s_{i,k} = L_f + x_{i,k} + l_{i,k}/2, \tag{4}$$

$$\frac{1}{WB} \cdot \sum_{i=1|i \in R_v}^{n} \sum_{k=1}^{c_i} (m_{i,k} \cdot g \cdot s_{i,k}) = F_{RA} \tag{5}$$

and

$$\sum_{i=1|i \in R_v}^{n} \sum_{k=1}^{c_i} (m_{i,k} \cdot g) - F_{RA} = F_{FA}. \tag{6}$$

The acting forces must be below the permissible ones, but also greater than zero to avoid uplifting:

$$F_{FA} \leq FA_{perm} \cdot g, \tag{7}$$

$$F_{RA} \leq RA_{perm} \cdot g, \tag{8}$$

$$F_{FA} \geq 0, \tag{9}$$

and

$$F_{RA} \geq 0. \tag{10}$$

As demonstrated in Krebs and Ehmke (2021), the constraint must be checked after each placement of an item since, an axle may become overloaded after unloading items.

4.6 Balanced loading (C10)

To prevent a reduction in vehicle stability, the load per vehicle half should be examined. Therefore, Pace et al. (2015) suggest that a percentage p of the vehicle capacity D is not exceeded. In the following, we introduce formulas for this approach.

In our implementation, the item's mass $m_{i,k}$ is assigned to the vehicle sides depending on its y-position ($y_{i,k}$). If an item lays entirely on the left side of the vehicle (see Fig. 10a), its total mass is assigned to the left vehicle side. The same is true for the opposite right side (see Fig. 10b). Otherwise, the mass of the item is distributed proportionally to the vehicle sides (see Fig. 10c). The sum of all assigned masses must not exceed a certain percentage p of the load capacity D. Consequently, the following must apply:

$$f(t) = \begin{cases} t & t > 0 \\ 0 & \text{else} \end{cases} \tag{11}$$

$$\sum_{i=1|i\in R_v}^{n} \sum_{k=1}^{c_i} \frac{m_{i,k}}{l_{i,k}} \cdot \left[f\left(\frac{W}{2} - y_{i,k}\right) - f\left(\frac{W}{2} - (y_{i,k} + w_{i,k})\right) \right] \leq D \cdot p \tag{12}$$

$$\sum_{i=1|i\in R_v}^{n} \sum_{k=1}^{c_i} \frac{m_{i,k}}{l_{i,k}} \cdot \left[f\left((y_{i,k} + w_{i,k}) - \frac{W}{2}\right) - f\left(y_{i,k} - \frac{W}{2}\right) \right] \leq D \cdot p \tag{13}$$

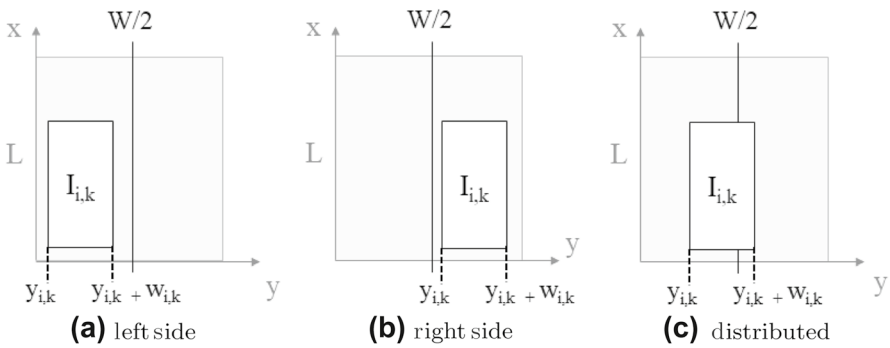


Fig. 10 Mass distribution according to the position of $I_{i,k}$

Formula 11 restricts the range to positive real values. It is used to assign the masses to the corresponding vehicle sides. The constraint for the left vehicle side is checked in Formula 12 and in 13 for the right one, respectively. Inside of the large square brackets, the position of the item with respect to the corresponding vehicle side is determined in order to assign the proportional mass.

5 Hybrid solution approach

We propose a hybrid solution approach consisting of a routing heuristic (adaptive large neighbourhood search) for creating routes and an embedded packing heuristic (deepest-bottom-left-fill algorithm), which optimizes the loading of the items of all customers of a route into the loading space of a vehicle. The packing heuristic generates feasible packing plans for the generated routes. This packing plan is created following a loading constraint set P, which determines the included loading constraints.

5.1 Routing heuristic

We use the routing algorithm as described in Koch et al. (2018), who modified the adaptive large neighbourhood search (ALNS) proposed by Ropke and Pisinger (2006). The algorithm by Koch et al. (2018) was developed for the 3L-VRPTW with Backhauls and is applied to the pure 3L-VRPTW in this paper, considering additional constraints. The general framework is shown in Alg. 1. The corresponding line number of the algorithms are given in square brackets. In general, a solution is feasible if all loading and routing constraints are obeyed except S3 so that the used vehicles could exceed the number of available vehicles.

Algorithm 1 Adaptive Large Neighbourhood Search

Input: Instance data, parameters
Output: best feasible solution s_{best}

```
construct initial solution s_init
s_best := s_init
s_curr := s_init
do
    select removal operator rem
    select insertion operator inst
    select number of customers to be removed n_rem
    determine next solution s_next := inst(rem(s_curr, n_rem))
    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_next
        end if
    end if
    if iter_p reached then
        update selection probabilities for insertion and removal heuristics
    end if
while one stopping criterion is not met
```

5.1.1 Initial solution

The initial solution s_{init} is constructed [1] with the savings heuristic developed by Clarke and Wright (1964). Hereby, all routing (except S3) and loading constraints are obeyed. Based on this feasible initial set of routes, the ALNS determines other feasible improved solutions.

5.1.2 Iteration

In each iteration of the ALNS, one removal *rem* and one insertion operator *inst* are randomly chosen [5-6]. These are used to generate the next solution s_{next} by removing a number of customers n_{rem} from the solution and reinserting them again [8]. The number of customers to be removed n_{rem} ($n_{min} \leq n_{rem} \leq n_{max}$) is determined randomly [7]. Then, it is checked whether the generated solution meets the routing constraints [9] described in Sect. 3. The packing procedure shown in the next subsection is called here.

5.1.3 Evaluation function

In order to evaluate different solutions and to lead the search, the following internal evaluation function is defined. The evaluation function f for a solution s giving total routing costs is described as follows:

$$f(s) = ttd(s) + pen_v \cdot max(0, v_{used} + |N_{miss}| - v_{max}) + \sum_{i \in N_{miss}}^{N_{miss}} (c_{0,i} + c_{i,0}), \quad (14)$$

where N_{miss} is a set containing all customers that have not been dispatched yet, v_{max} is the maximal number of available vehicles and v_{used} the number of used vehicles. Each customer i, which is not yet dispatched ($i \in N_{miss}$), is assigned to one vehicle (round-trip) even if this leads to an exceedence of the number of used vehicles. The penalty term pen_v is used to achieve a reduction of used vehicles v_{used}. Additionally, the total travel distance $ttd(s)$ for a solution s is respected.

5.1.4 Solution acceptance

A solution is regarded better the smaller its evaluation function value is. A better and feasible solution is always accepted. A worse solution may be accepted [9] according to an acceptance probability which depends on a simulated annealing heuristic proposed by Kirkpatrick et al. (1983). In particular, the acceptance probability is adapted to the annealing process with a geometric cooling schedule. The best solution s_{best} is updated [13] if it has a superior evaluation function value relative to the current solution s_{curr} [12].

Table 4 Overview removal operators

Neighbourhood operators	Description
Shaw	Removes related customers w.r.t. distance, demand, time windows
Random	Removes random customers
Worst	Removes customers increasing the total routing costs the most
Cluster	Divides a random tour into two clusters and randomly removes one of the cluster
Neighbour graph	Removes customers increasing the average distance of a tour
Overlap	Removes customers leading to intersection of two tours
Inner route	Removes a tour which is completely surrounded by another and splits the surrounding tour into two
Intersection	Removes customers leading to intersections within a tour
Tour pair	Removes two intersecting tours

Table 5 Overview insertion operators

Neighbourhood operators	Description
Greedy	Inserts customers iteratively so that an increase of routing costs is minimal
Regret-2	Inserts customers iteratively so that the maximal difference of routing costs for the best and the second best insertion in different tours is achieved
Regret-3	Inserts customers iteratively so that the sum of two differences of routing costs is maximal. The first difference is the routing cost for the best and the second best insertion in different tours, while the second difference results from the best and the third best insertion in different tours

5.1.5 Removal and insertion operators

Table 4 shows nine removal operators and Table 5 summarises the three insertion approaches used in this paper. We use the removal and insertion operators as described and evaluated in Koch et al. (2018).

After a defined number of iterations $iter_p$, the selection probabilities for the removal and insertion operators are adjusted [16-18] according to their improvement of the solution. This is described in detail in the following section.

5.1.6 Operator selection and probability adaption

The selection of the operators is accomplished by means of the roulette wheel selection principle. Hereby, the probability to select one operator op is defined by their weighting wg_{op}. Initially, all operators have the same selection probability ($wg_{op} = 1$).

The number of iterations is counted, in which the operator *op*

- is selected ($counter_{op}$),
- is selected and led to a new best solution ($iter_{best_{op}}$),
- is selected and improved the current solution ($iter_{impr_{op}}$),
- is selected and led to a worse but not yet accepted solution or a solution as good as the current solution ($iter_{e_{op}}$).

After a certain number of iterations $iter_p$, the success of the operator is evaluated and described by $score_{op}$, which is calculated as follows:

$$score_{op} = iter_{best_{op}} \cdot \omega_{best} + iter_{impr_{op}} \cdot \omega_{impr} + iter_{e_{op}} \cdot \omega_e. \tag{15}$$

Hereby, ω_{best}, ω_{impr} and ω_e are coefficients.

Then, the new weighting wg_{op} can be calculated. A reaction factor r regulates the influence of the adaptions:

$$wg_{op} = wg_{op} \cdot (1 - r) + r \cdot \frac{score_{op}}{counter_{op}} \tag{16}$$

Moreover, $counter_{op}$, $iter_{best_{op}}$, $iter_{impr_{op}}$ and $iter_{e_{op}}$ are reset to zero.

5.1.7 Stopping criteria

If one of the following stopping criteria is met [19], the heuristic terminates, and the current best known solution is given:

- Number of total iterations $iter_{max}$;
- Number of iterations without improvement $iter_{wimpr}$;
- Calculation time limit t_{max}.

5.2 Packing heuristic

As packing heuristic, we use the same approach as in Krebs and Ehmke (2021), which is based on the deepest-bottom-left-fill (DBLF) algorithm proposed by Karabulut and İnceoğlu (2005). The algorithm is detailed in Alg. 2. The basic concept is to place the items as far as possible to the back (first priority), to the bottom (second priority) and to the left (third priority) of the loading space. The available free spaces in the vehicle's loading space are stored in a list.

In the following, the point of origin of a Cartesian coordinate system is assumed to be located in the deepest, bottom, leftmost point of the loading space. The driver's cab is located behind it accordingly. The length, width and height of the loading space are parallel to the x-, y- and z-axes. The placement of an item $I_{i,k}$ is defined by $(x_{i,k}, y_{i,k}, z_{i,k})$ of the corner which is closest to the point of origin.

Before starting the packing process, the items of each customer are sorted by means of the following priorities:

1. fragility flag $f_{i,k}$ (non-fragile first)
2. volume (larger volume first)
3. length $l_{i,k}$ (longer first)
4. width $w_{i,k}$ (wider first).

Then, the items are added to the packing sequence *IS* reversed to the customer's visiting order [1]. Let *S* be the set of unique cuboids representing available free spaces for placing items. Initially, the set consists of one potential space, which corresponds to the entire loading space [2]. Therefore, the first item of the packing sequence is placed in the origin. The potential spaces of the set *S* are always sorted based on the DBL-rule [10]. Thus, an item is placed in the deepest, bottom, leftmost point of the space. For each item I_c ($c = 1, ..., |IS|$), a feasible placement is determined [3]. For this purpose, each space *sp* of the set is tested as possible item position [5] until a feasible position is found obeying all loading constraints of the loading set *P* [7-8]. In comparison to Karabulut and İnceoğlu (2005), the set *S* does not contain all available placements inside the loading space. Rather, three new spaces (front, right, top) are created based on the feasible item placement [9].

The front (right, top) space is defined by the item's front (right, top) edge and either the door (wall, ceiling) or the nearest item in front (rightmost, topmost) of the item. Then, the minimum and maximum values for the y-(x, y) and z-axis (z, x) limited by the loading space or other items are searched. Fig. 11 shows these three created spaces exemplary based on item $I_{3,1}$. Additional three spaces are created if they are unique: Another front and right space, where the minimum z-value represents the bottom edge of item I_c, and another top space, where the minimum x-value is the deepest edge of item I_c. The new spaces (front, right, top) are included in the set.

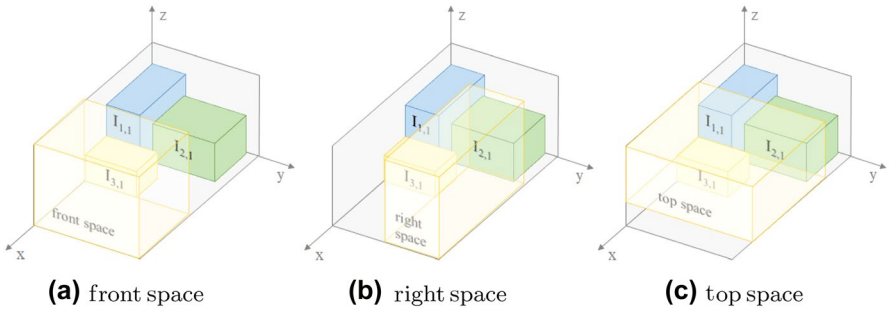


Fig. 11 New spaces based on $I_{3,1}$

Algorithm 2 Deepest-Bottom-Left-Fill with Spaces

Input: Instance data
Output: Feasibility, Packing Plan PP_v

```
initialize sorted sequence of items IS
initialize set of unique available spaces S
for each item I_c ∈ IS do
    for each space sp ∈ S do
        for each permitted orientation do
            if item I_c fits in space sp then
                if placement is feasible w.r.t. the constraint set P then
                    save placement for I_c
                    create new spaces
                    sort spaces based on DBL
                    erase space sp
                    get smallest dimensions l_min and h_min of unplaced items ∈ IS
                    for each space si ∈ S do
                        update space si
                        if si too small then
                            erase space si
                        end if
                    end for
                    break
                end if
            end if
        end for
    end for
    if no feasible position found then
        return false
    end if
end for
return true
```

After each feasible placement of an item I_c, the available spaces are updated [13–14], which means that all available spaces are checked w.r.t. an intersection with item I_c. If one or more spaces intersect with item I_c, then these spaces are decreased so that no intersection occurs. Therefore, if an item can be placed within an available space, it is guaranteed that the item does not overlap with other items or with the vehicle's walls (geometry constraint (C1)). In contrast to the approach by Karabulut and İnceoğlu (2005), an overlapping check between each item is not necessary for this approach, which improves the performance. The used space is removed from the set [11]. To increase the efficiency of the packing heuristic and to reduce the number of spaces in the set, only spaces which are large and high enough for the smallest dimensions of any unplaced item of the route are inserted in the set S. Therefore, the shortest length or width l_{min} and height h_{min} of any unplaced item of the route are searched [12]. Due to the permitted rotations, only the two measures l_{min} and h_{min} are relevant. If the length or height of any space in the set is smaller than l_{min} or h_{min}, the space is removed from the set [15-17]. Then, a placement for the next item is searched [19].

If no feasible position for the item can be found, the route is revised, and a new one must be searched by the ALNS [24-26].

6 Computational studies

In this section, we investigate the solution quality and the performance of the hybrid algorithm in the context of advanced loading constraints. We use well-known instance sets and investigate the impact of the proposed loading constraints on the objective values by means of a new instance set. All results along with detailed packing plans are available via https://github.com/CorinnaKrebs/Results.

The hybrid algorithm is implemented in C++ as single-core, x64-application and is compiled using the GCC version 4.8.3, compiler. The experiments were executed on a High Performance Cluster, Haswell-16-Core with 2.6 GHz.

6.1 Parameters

The parameters for the loading constraints (see Sect. 4) and for the routing heuristic (see Sect. 5) are listed in Table 6. Regarding the parameters for the routing heuristic, we performed a preliminary study to tune the parameters. As the evaluation showed, the best results were obtained by the parameters as described in Koch et al. (2018) and therefore, these parameters were set. The parameters for the loading constraints are those used in the literature so far.

6.2 Instances

For our computational study, we use the instance sets by Moura and Oliveira (2009), Ceschia et al. (2013) and Zhang et al. (2017). Moreover, a new instance set consisting of 600 instances is created. The characteristics of the instance sets are shown in Table 7. Our new instance set is available via https://doi.org/10.24352/UB.OVGU-2020-139.

The instances vary in the number of customers, items and item types. They either have 20, 60 or 100 customers, which demand either 200 or 400 items in total. These items differ in their homogeneity: Either there are only three item types (very homogeneous), 10 item types or 100 different item types (very heterogeneous). For realistic item masses, we analysed 12,000 products from a Swedish furniture company which offers products of different categories among other housewares, decorative articles and groceries. The densities of these products vary mainly between 0.5 and 1.5 kg/dm^3. So the densities for the items are assigned by choosing a value randomly within this interval. Thereby, it is considered that the total mass for one customer is less than the vehicle load capacity D, since a customer can only be served by a vehicle (see constraints S1 and C4).

The fragility flag is set randomly to the items, where approx. 30% are fragile. To define the parameters for the load bearing strength, the formula by Ratcliff and Bischoff (1998) is used. For each item, a value r is determined depending on the fragility flag: If an item is fragile, then the value r is randomly chosen in the interval [1.0,

Table 6 Routing and loading parameters

Parameter	Usage	Description	Value
$iter_{max}$	Stopping criterion	Maximal number of iterations	25,000
$iter_{wimpr}$	Stopping criterion	Maximal number of iterations without improvement	8,000
t_{max}	Stopping criterion	Time limit [min]	60
$iter_p$	ALNS	Number of iterations for updating probabilities for removal and insertion operators	100
ω_{best}	ALNS	Coefficient for determination of the operator score, weighting the influence of finding new best solutions	50
ω_{impr}	ALNS	Coefficient for determination of the operator score, weighting the influence of finding improved solutions	10
ω_e	ALNS	Coefficient for determination of the operator score, weighting the influence of finding worse, not yet accepted solutions or solutions as good as the current solution	5
r	ALNS	Reaction factor	0.8
n_{min}	ALNS	Number of minimal customers to be removed from a route	$0.04n$
n_{max}	ALNS	Number of maximal customers to be removed from a route	$0.4n$
d_{max}	Evaluation Function	Maximal distance between two customers in instance	$max_{i,j \in N} d_{i,j}$
pen_v	Evaluation Function	Penalty term for each surplus vehicle	$10 \cdot d_{max}$
α	Vertical stability	Minimal supporting ratio	0.75, 1.00
λ	Reachability	Minimal distance to reach an item [dm]	5
p	Balanced loading	Maximal mass ratio of vehicle's capacity for one vehicle side	0.7

Table 7 Overview of instance sets

Author	Problem	#	n	m
Ceschia et al. (2013)	3L-CVRP	13	[13, 129]	[254, 8060]
Moura and Oliveira (2009)	3L-VRPTW	46	25	1050, 1550
Zhang et al. (2017)	3L-VRPTW	27	[15, 100]	[26, 199]
This paper	3L-VRPTW	600	20, 60, 100	200, 400

2.0]. Otherwise, r lays in the interval [1.0, 5.0]. Then, over all items, the value $max \frac{m_{i,k}}{m_{i,k} \cdot l_{i,k}}$ is searched and for each item multiplied by r.

To ensure a realistic proportion between vehicle load capacity and axle weights, parameters from the two-axle truck ML180 by IVECO were chosen, which has a maximum payload of 12,595 kg. Then, a proportional factor pr was calculated on the basis of the vehicle load capacity D of an instance and the maximum payload of the IVECO truck. Thus, the following applies: $pr = \frac{D}{12595}$. The axle weights for the front and the rear axle were then proportionally scaled on the basis of pr.

6.3 Evaluation of hybrid algorithm

This section deals with the evaluation of the hybrid algorithm concerning its solution quality and performance. Hereby, we use our instance set and the benchmark instances by Ceschia et al. (2013), Zhang et al. (2017), and Moura and Oliveira (2009). Every instance is tested five times. We present summarized results. The more detailed results are presented in the "appendix" and are available via https://github.com/CorinnaKrebs/Results. Note again that smaller objective values (v_{used} and ttd) represent better results.

In Table 8, we compare the received best and average results based on the *basic constraint set* used in Gendreau et al. (2006). On average, the difference between best and average for the number of vehicles is 1.42%, for the total travel distance only 0.42%. The results show a tendency that the more difficult the instances are (more customers or items), the higher the deviation between best and average results for the objective values. Therefore, we see potential for improvement as the hybrid algorithm should achieve an average deviation of only 1% at most.

The following Table 9 presents the best results per instance set.

Concerning the Ceschia et al. (2013) instances, some of the instances require split delivery or feature a heterogeneous vehicle fleet. Excluding these instances, seven instances remain. Ceschia et al. (2013) have a maximum time limit varying between 300 and 10000 seconds. In contrast, the calculation time limit for our computational tests is only 3600 seconds. The results are based on the *basic constraint set*. The hybrid algorithm achieves clearly better results than the benchmark for nearly all instances (except SD-CSS04). On average, 19.33% less vehicles are used and the total travel distance decreases by 11.87%. Moreover, a shorter calculation time is required.

Table 8 Comparison best and average results for P1, our instances

		Best			Average			Difference		
		sum v_{used}	sum *ttd*	avg. *time [s]*	sum v_{used}	sum *ttd*	avg. *time [s]*	v_{used}	*ttd*	*time*
n										
	20	503.00	52,742.06	2,023.92	505.80	52,837.07	2,035.13	0.56%	0.18%	0.55%
	60	3,506.00	291,080.57	2,003.48	3,548.40	292,496.96	1,996.96	1.21%	0.49%	-0.33%
	100	4,007.00	364,642.68	2,331.85	4,075.60	366,081.91	2,328.51	1.71%	0.39%	-0.14%
m										
	200	3,052.00	300,443.12	1,702.83	3,089.60	301,415.49	1,699.53	1.23%	0.32%	-0.19%
	400	4,964.00	408,022.19	2,575.00	5,040.20	410,000.45	2,574.90	1.54%	0.48%	0.00%
Item types										
	3	2,523.00	225,549.08	1,725.60	2,550.00	226,265.32	1,729.26	1.07%	0.32%	0.21%
	10	2,671.00	236,649.38	2,311.98	2,717.00	237,720.68	2,304.15	1.72%	0.45%	-0.34%
	100	2,822.00	246,266.85	2,379.17	2,862.80	247,429.94	2,378.23	1.45%	0.47%	-0.04%
Total		8,016.00	708,465.31	2,138.92	8,129.80	711,415.94	2,137.22	1.42%	0.42%	-0.08%

Table 9 Summarized best results for instance sets

	Benchmark results		Our best results				
	Sum v_{used}	Sum *ttd*	Sum v_{used}	Sum *ttd*	Avg. *time* [s]	Diff. v_{used}	Diff. *ttd*
Ceschia et al. (2013)	150	122,678.60	121	108,110.62	3,600.00	– 19.33%	– 11.87%
Zhang et al. (2017)	383	26,074.94	294	21,039.00	469.70	– 23.24%	– 19.31%
	Sum v_{used}	Avg. *ttd*	Sum v_{used}	Avg. *ttd*	Avg. *time* [s]	Diff. v_{used}	Diff. *ttd*
Moura and Oliveira (2009)	247	548.5	297	536.66	3258.14	20.24%	– 2.16%

For the Zhang et al. (2017) instances tested with the *basic constraint set*, the hybrid algorithm achieves a reduction of 23.24% used vehicles and a reduction of the total travel distance by 19.31%, on average. Moreover, the presented hybrid algorithm achieves the results with half of the calculation times compared to the benchmark. In case of the Moura and Oliveira (2009) instances, the instances do not provide any item masses, vehicle load capacities or fragility flags. Moreover, fully support of items is required ($\alpha = 1$) and we only rotate the items along the length-width plane. The currently best-known results are received by Reil et al. (2018). In comparison to the benchmark, the number of used vehicles increases by 20.24%, while the total travel distance decreases by 2.16%. The reason for these results is that we cannot use all rotation possibilities and only a small amount of iterations are conducted due to the high number of items per vehicle. We see potential for improvements of our hybrid algorithm to be able to compete with these instances.

To summarize, the hybrid algorithm finds new best results for two of three benchmark instances. In case of a high number of items per vehicle, the hybrid algorithm is not competitive so that we need to address this in our further research.

6.4 Evaluation of loading constraints

The following subsections analyse the impact of the different loading constraints on the objective values.

6.4.1 Constraint sets

In order to evaluate the impact of the new loading constraints in a systematical way, one new constraint is either replaced or added based on the *basic constraint set* P1. The last constraint set is a combination of the most restrictive ones. Table 10 shows the loading constraints as considered in each set. Details are as follows:

Table 10 Overview of constraints sets

Constraints		Sets P									
		Basic	Replacing					Adding			Combination
		1	2	3	4	5	6	7	8	9	10
C1	Geometry	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
C2	Orthogonality	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
C3	Rotation	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
C4	Load capacity	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
C5a	LIFO	✓	□	✓	✓	✓	✓	✓	✓	✓	✓
C5b	MLIFO		☑								
C6a	Minimal supporting area	✓	✓	□	□	✓	✓	✓	✓	✓	□
C6b1	Multiple overhanging			☑							
C6b2	Top overhanging				☑						☑
C7a	Fragility	✓	✓	✓	✓	□	□	✓	✓	✓	□
C7b1	LBS—simplified selection					☑					
C7b2	LBS—complete selection						☑				☑
C8	Reachability							☑			☑
C9	Axle weights								☑		☑
C10	Balanced loading									☑	☑

✓: included in P1; □: removed compared to P1; ☑: additional compared to P1

1. Replacing: The constraint sets P2–P6 are created by replacing one definition or implementation variant with another.
2. Adding: The constraint sets P7–P9 are generated by adding further loading constraints to P1.
3. Combination: The constraint set P10 is a combination of replacing and adding of loading constraints.

6.4.2 Results

For the analysis of the loading constraints, we use our new instance set to enable comparison concerning the number of customers (n), items (m) and item types. Every instance is tested five times for each constraint set. In sum, our analysis is based on 30,000 results (600 instances, 10 constraint sets, 5 runs). In the following Table 11, we report the average results and calculate the percentage difference to the basic constraint set P1. Note again that smaller objective values (v_{used} and ttd) represent better results.

The impact of the MLIFO constraint (C5b) is evaluated by set P2. Compared to the LIFO constraint (C5a), on average, the objective values decrease slightly by around 0.2%. The solution space of the MLIFO constraint (C5b) is larger; contrasting our expectations, a significant influence of an increasing number of customers, items or item types on the objective values is not evident, though. **To be more flexible in the handling of items, we recommend to rely further on the LIFO (C5a) constraint.**

The robust stability (sets P3 and P4) reduces the solution space due to its more stable definition in comparison to the minimal supporting area definition. Thus, the constraints lead to a notable increase of the objective values and the calculation time. In case of the multiple overhanging constraint, the number of used vehicles rise by 10.80%, the total travel distance by 8.27%, on average. Since the top overhanging constraint is more restrictive, the number of used vehicles increases by additional 4.16% points, the total travel distance by 2.66% points, on average. For both variants, the calculation time increases by around 60%. Moreover, the more heterogeneous the items are (more item types), the more the objective values rise, since homogeneous item stacks items do not overhang and therefore, the constraint is fulfilled. Another aspect is the calculation time. In case of instances with 200 items, the calculation time increases by around 97%. In contrast, the increase for instances with 400 items is only around 37%. The explanation is as follows: In general, the higher the number of items, the higher the calculation time and the smaller the difference to the maximum calculation time. In case of the robust stability constraints, the maximum calculation time is exploited for most instances and therefore also for those, which had a small calculation time for the basic set P1. **We recommend using the top overhanging constraint since its more stable definition.**

The constraint sets P5 and P6 deal with the impact of the load bearing strength constraint. In general, the simplified and the complete selection variants lead to comparable objective values. On average, the number of used vehicles increases by

approx. 3.2% (v_{used}), the total travel distance by approx. 2.6% for both approaches. The objective values of some instances are even smaller since the fragility constraint (C7a) is more restrictive than the load bearing strength constraints in this case. Since the load of items is calculated for the entire stack starting from the last placed item to the vehicle floor, the calculation time increases rapidly due to the algorithmic complexity (on average by around 29%) and also the objective values increase with the number of items. Interestingly, the objective values decrease with the number of customers. An explanation could be that with a lower number of customers, a higher number of items per customer is demanded. Since all items of a customer have to be packed into a vehicle, more items are stacked on top of each other, so that the limit values for the LBS are reached. Furthermore, our results show that a higher number of item types has a positive effect on the objective values. The reason is that with homogeneous item stacks, the load is distributed over fewer items. **Since both variants lead to similar results and due to the fact that the complete selection variant is more realistic, we recommend using the complete selection.**

The reachability constraint (C8), evaluated by set P7, leads to an increase of the number of used vehicles by 4.06% and the total travel distance by 2.80%, on average. However, the constraint has almost no effect on the objective values for half of the instances. A higher number of items or item types leads to an increase of the objective values by some per cent points, because in case of heterogeneous items or of overall more items, it is more likely that the items block the way. **As the constraint has rather small impacts and avoids unnecessary rearrangements of items during unloading, it is recommended to take it into account in practically oriented VRP computations.**

The sets P8 and P9 deal with the effects of distributed masses. The axle weights constraint (C9) distributes the masses in the vehicle along the x-axis, while the load is balanced along the y-axis in the balanced loading constraint (C10). For the majority of instances, the objective values remain unchanged or increase by only a few per cent. On average, the number of used vehicles increase by around 2.9%, the total travel distance by approx. 1.8%. Moreover, there is a positive effect on the calculation time, which is reduced by around 20%. A correlation between objective values and the number of customers, items and item types is not apparent. **Due to the small impact on the objective values, the positive effects on the calculation time and the great safety relevance in traffic, considering both constraints makes sense if the appropriate information is available.**

Since P10 contains all new complex constraints, the objective values clearly deteriorate, whereby the number of used vehicles increases more (24.42%) than the total travel distance (17.15%), on average. A comparison with the previous results reveals that a combination of several loading constraints leads to a deterioration of the objective values but not to the extent that the sum of the deteriorations would result from individually investigated constraints. The same applies for the calculation time.

Finally, regarding the results of all loading constraints, a correlation between an increasing number of customers or items and the objective values is not evident.

Table 11 Deviation to P1 per constraint set, average results

		n			*m*		*Item types*			Total
		20	60	100	200	400	3	10	100	
P1 –	sum v_{used}	505.80	3,548.40	4,075.60	3,089.60	5,040.20	2,550.00	2,717.00	2,862.80	8,129.80
Basic	sum *ttd*	52,837.07	292,496.96	366,081.91	301,415.49	410,000.45	226,265.32	237,720.68	247,429.94	711,415.94
	avg. *time* [s]	2,035.13	1,996.96	2,328.51	1,699.53	2,574.90	1,729.26	2,304.15	2,378.23	2,137.22
P2 –	diff. v_{used}	0.24%	−0.32%	−0.20%	−0.28%	−0.19%	−0.19%	−0.22%	−0.26%	−0.22%
MLIFO	diff. *ttd*	0.14%	−0.22%	−0.01%	−0.08%	−0.09%	0.05%	−0.17%	−0.13%	−0.09%
	diff. *time*	0.70%	1.76%	0.32%	1.80%	0.36%	1.11%	0.20%	1.51%	0.93%
P3 –	diff. v_{used}	1.70%	9.10%	13.40%	12.33%	9.86%	6.60%	11.65%	13.73%	10.80%
Multiple	diff. *ttd*	1.64%	8.40%	9.13%	8.97%	7.76%	5.47%	8.65%	10.47%	8.27%
Overhanging	diff. *time*	41.83%	78.24%	54.61%	96.93%	37.29%	96.78%	47.13%	48.44%	61.01%
P4 –	diff. v_{used}	7.55%	13.43%	17.21%	15.56%	14.60%	7.24%	14.65%	22.14%	14.96%
Top	diff. *ttd*	4.01%	11.20%	11.71%	10.86%	10.98%	5.79%	10.96%	15.60%	10.93%
Overhanging	diff. *time*	44.00%	78.16%	54.48%	97.86%	37.22%	96.94%	47.07%	49.25%	61.33%
P5 –	diff. v_{used}	8.19%	2.90%	3.15%	2.50%	3.88%	6.64%	3.26%	0.52%	3.35%
LBS	diff. *ttd*	4.17%	2.39%	2.73%	1.90%	3.28%	4.67%	2.48%	1.09%	2.69%
Simple	diff. *time*	33.68%	35.50%	23.02%	39.23%	23.43%	51.94%	22.26%	20.77%	29.71%
P6 –	diff. v_{used}	6.84%	2.69%	3.05%	2.30%	3.63%	6.20%	2.89%	0.62%	3.13%
LBS	diff. *ttd*	3.92%	2.14%	2.52%	1.76%	2.98%	4.43%	2.19%	0.93%	2.47%
Complete	diff. *time*	33.88%	35.82%	22.74%	38.68%	23.86%	52.28%	22.02%	20.86%	29.75%
P7 –	diff. v_{used}	1.86%	4.74%	3.73%	3.35%	4.49%	2.66%	4.22%	5.15%	4.06%
Reachability	diff. *ttd*	1.21%	3.17%	2.73%	2.14%	3.28%	1.91%	2.88%	3.53%	2.80%
	diff. *time*	7.26%	7.48%	5.04%	9.89%	4.06%	6.93%	3.87%	8.40%	6.38%
P8 –	diff. v_{used}	1.23%	4.08%	2.05%	2.80%	2.93%	2.79%	2.94%	2.91%	2.88%
Axle Weights	diff. *ttd*	0.61%	2.95%	1.18%	1.90%	1.84%	1.88%	1.89%	1.84%	1.87%
	diff. *time*	−42.53%	−24.94%	−7.83%	−31.01%	−14.11%	−23.71%	−20.65%	−18.92%	−20.83%

Table 11 (continued)

		n			*m*		*Item types*			Total
		20	60	100	200	400	3	10	100	
P9 –	diff. v_{used}	1.23%	4.22%	2.08%	3.05%	2.90%	2.87%	2.80%	3.19%	2.96%
Balanced	diff. *ttd*	0.61%	2.78%	1.11%	1.86%	1.68%	1.88%	1.56%	1.83%	1.76%
Load	diff. *time*	−42.86%	−24.08%	−6.61%	−29.43%	−13.85%	−25.13%	−18.74%	−17.61%	−20.04%
P10 –	diff. v_{used}	23.13%	23.83%	25.10%	23.80%	24.80%	20.93%	23.67%	28.24%	24.42%
Combination	diff. *ttd*	10.52%	17.99%	17.43%	15.50%	18.36%	14.44%	16.88%	19.88%	17.15%
	diff. *time*	54.23%	77.67%	54.45%	101.39%	37.80%	99.29%	50.35%	49.10%	63.08%

However, an increase of the number of item types and therefore an increase of the degree of heterogeneity tends to be correlated with an increase of the objective values except for the load bearing strength constraints, where the load can be better distributed along heterogeneous items.

7 Conclusions and future work

This paper continues the research on the combined Vehicle Routing Problem and 3D Loading (3L-CVRP) introduced by Gendreau et al. (2006) and the extended problem with the consideration of Time Windows (3L-VRPTW). In our implementation, the possible placement of items is represented by free spaces inside the vehicle's loading space improving the performance of the algorithm. For a more realistic modelling, new loading constraints are introduced. Since the common definition of stability leads to unstable stacks, the robust stability is investigated by means of two implementation variants. In the first one, items of a stack are allowed to overhang when respecting a minimal supporting area at any height (multiple overhanging). In the second variant, only the topmost item of a stack is allowed to overhang (top overhanging). Moreover, instead of a simple fragility flag grouping items in fragile and non-fragile ones, the load per area unit for each item is considered. For this load bearing strength constraint, also two implementation variants (simplified and complete) are investigated. Additionally, constraints regarding the reachability of items, the axle weights and the balanced loading inside the vehicle are considered as well as the Manual LIFO by Tarantilis et al. (2009). The solution quality and the performance of our hybrid algorithm are evaluated by using well-known instances by Moura and Oliveira (2009), Ceschia et al. (2013) and Zhang et al. (2017). For the latter two instance sets, the presented algorithm performs better than the benchmarks.

The impact of the loading constraints on the objective values (number of used vehicles and total travel distance) is tested by 600 new instances varying in the number of customers, items and item types. In most cases, the Manual LIFO constraint has no influence on the objective values. Both variants for the load bearing strength constraint lead to comparable results. Therefore, the usage of the realistic "complete" variant instead of the simplified one is recommended. In case of the robust stability, we recommend using the top overhanging variant since it achieves higher stability of the stacks. The axle weights and the balanced loading constraints only lead to small increases of the objective values and even decrease the calculation time. Since they increase the vehicle stability and thus the safety, we recommend to investigate these further in future research. The same applies to the reachability constraint, which prevents unnecessary rearrangements during unloading. Furthermore, our investigations showed that when combining complex constraints, the results deteriorate, but not to the extent that the sum of the deterioration would result from

individually investigated constraints. As future work, we suggest to improve the performance of the hybrid algorithm and of the complex loading constraints. Furthermore, we plan to determine the influence of the individual neighbourhood operators on the results as well as impact of instance features on the packing algorithm.

Appendix

See Tables 12, 13, 14 and 15.

Table 12 Results for Ceschia et al. (2013) instances

Instance	Ceschia et al. (2013)		Our results					
	Best		Best			Average		
	v_{used}	*ttd*	v_{used}	*ttd*	*time* [s]	v_{used}	*ttd*	*time* [s]
SD-CSS1	5	5,708.60	5	5,152.2	3,600	5	5,152.2	3,600
SD-CSS2	13	12,033.2	13	11,865.7	3,600	13	11,866.8	3,600
SD-CSS4	12	11,398.6	12	11,470.4	3,600	12	11,794.8	3,600
SD-CSS9	23	17,724.8	17	13,789.5	3,600	17	13,891.5	3,600
SD-CSS10	18	12,945.9	9	10,103.9	3,600	9	10,269.5	3,600
SD-CSS12	48	34,807.3	45	37,458.4	3,600	46.8	37,274.6	3,600
SD-CSS13	31	28,060.2	20	18,270.5	3,600	20	18,346.1	3,600
Total	150	122,678.6	121	108,110.6	25,200	122.8	108,595.6	25,200

Table 13 Results for Moura and Oliveira (2009) instances

	Reil et al. (2018)		Our results					
	Best		Best			Average		
	Sum v_{used}	Avg. *ttd*	Sum v_{used}	Avg. *ttd*	Avg. *time* [s]	Sum v_{used}	Avg. *ttd*	Avg. *time* [s]
GI								
I1	62	545.3	70	536.28	2,847.94	72.8	537.72	2,870.75
I2	44	525.0	56	498.73	3,600.00	58.6	503.45	3,600.00
GII								
I1	75	577.9	93	573.17	3,041.59	93	573.57	3,063.70
I2	66	543.5	78	535.19	3,600.00	78	535.69	3,600.00
Total	247	548.5	297	536.66	3,258.14	302.4	538.39	3,269.86

Table 14 Results for Zhang et al. (2017) instances

Instance	Zhang et al. (2017)					Our results					
	Best		Average			Best			Average		
	v_{used}	*ttd*	v_{used}	*ttd*	*time* [s]	v_{used}	*ttd*	*time* [s]	v_{used}	*ttd*	*time* [s]
VRPTWP01	5	323.33	5	322.33	352.10	4	245.44	3.95	4	245.44	4.13
VRPTWP02	5	295.29	5	299.89	232.28	5	276.64	0.49	5	276.64	0.56
VRPTWP03	5	303.60	5	303.60	430.27	4	274.55	30.01	4.4	286.23	19.92
VRPTWP04	6	380.19	6	380.80	371.88	6	336.79	3.22	6	336.79	3.28
VRPTWP05	7	416.35	7	417.66	545.72	6	345.89	17.81	6	346.63	14.78
VRPTWP06	7	408.99	7	409.09	306.38	6	374.22	2.29	6	374.81	2.15
VRPTWP07	7	407.91	7	409.89	604.70	5	324.29	22.06	5	324.29	23.87
VRPTWP08	7	425.90	8	425.90	637.32	6	320.75	20.80	6	320.75	23.42
VRPTWP09	8	530.50	10	532.63	549.60	8	476.80	10.08	8	476.80	12.98
VRPTWP10	10	668.74	11	669.11	956.40	7	487.60	58.50	7	502.98	56.85
VRPTWP11	11	619.34	9	626.27	1,032.98	7	493.58	114.46	7	495.04	113.10
VRPTWP12	9	688.60	10.4	690.10	671.37	9	575.04	14.76	9	575.04	15.45
VRPTWP13	11	626.72	8.4	630.43	1,347.24	6	452.05	467.32	6	455.65	466.71
VRPTWP14	8	765.37	12	766.78	1,221.68	8	550.16	89.08	8	550.16	112.32
VRPTWP15	12	713.23	10.6	715.72	1,091.86	8	527.62	98.78	8	534.01	119.26
VRPTWP16	11	746.67	11.8	748.15	429.32	11	693.92	8.49	11	693.92	9.08
VRPTWP17	15	994.28	15.2	997.27	484.99	14	951.11	17.31	14	953.94	17.82
VRPTWP18	18	1,206.51	18	1,207.60	636.18	11	1,052.43	31.63	11.4	1,028.11	31.17
VRPTWP19	16	1,211.99	15.6	1,217.63	783.59	12	971.43	127.27	12	972.59	122.97
VRPTWP20	24	1,644.56	23.2	1,652.96	1,512.11	17	1,311.32	765.60	17	1,322.19	769.84
VRPTWP21	23	1,603.88	22	1,612.42	2,174.74	16	1,189.80	1,943.54	16.2	1,203.65	1,887.20
VRPTWP22	26	1,811.19	26.2	1,817.75	2,170.98	18	1,466.27	305.06	18	1,468.57	303.27
VRPTWP23	24	1,654.13	24	1,675.25	1,950.79	17	1,325.73	690.63	17	1,339.59	687.80

Table 14 (continued)

Instance	Zhang et al. (2017)					Our results					
	Best		Average			Best			Average		
	v_{used}	*ttd*	v_{used}	*ttd*	*time* [s]	v_{used}	*ttd*	*time* [s]	v_{used}	*ttd*	*time* [s]
VRPTWP24	21	1,644.55	21.4	1,649.61	1,745.54	16	1,289.15	708.50	16	1,307.07	704.16
VRPTWP25	27	1,836.02	26.8	1,856.52	2,877.48	20	1,441.59	3,600.00	20.4	1,442.39	3,600.00
VRPTWP26	32	2,144.47	32	2,160.37	3,250.49	24	1,642.74	1,455.63	24	1,650.85	1,380.11
VRPTWP27	28	2,002.63	29.6	2,030.71	3,037.26	23	1,642.09	2,074.60	23	1,656.40	2,097.34
Total	383	26,074.94	387.2	26,226.44	31,405.25	294	21,039.00	12,681.87	295.4	21,140.53	12,599.54

Table 15 Average results per constraint set, our instances

		m					*Item types*			Total
		20	60	100	200	400	3	10	100	
P1 –	sum v_{used}	505.80	3,548.40	4,075.60	3,089.60	5,040.20	2,550.00	2,717.00	2,862.80	8,129.80
Basic	sum *ttd*	52,837.07	292,496.96	366,081.91	301,415.49	410,000.45	226,265.32	237,720.68	247,429.94	711,415.94
	avg. *time* [s]	2,035.13	1,996.96	2,328.51	1,699.53	2,574.90	1,729.26	2,304.15	2,378.23	2,137.22
P2 –	sum v_{used}	507.00	3,537.00	4,067.60	3,080.80	5,030.80	2,545.20	2,711.00	2,855.40	8,111.60
MLIFO	sum *ttd*	52,909.33	291,845.45	366,036.92	301,166.90	409,624.80	226,369.84	237,307.44	247,114.42	710,791.70
	avg. *time* [s]	2,049.45	2,032.05	2,336.01	1,730.12	2,584.11	1,748.50	2,308.73	2,414.11	2,157.11
P3 –	sum v_{used}	514.40	3,871.40	4,621.80	3,470.60	5,537.00	2,718.40	3,033.40	3,255.80	9,007.60
Multiple	sum *ttd*	53,704.01	317,066.12	399,494.94	328,465.00	441,800.07	238,634.45	258,294.10	273,336.52	770,265.07
Overhanging	avg. *time* [s]	2,886.44	3,559.43	3,600.00	3,346.92	3,535.20	3,402.80	3,390.13	3,530.24	3,441.06
P4 –	sum v_{used}	544.00	4,025.00	4,777.20	3,570.20	5,776.00	2,734.60	3,115.00	3,496.60	9,346.20
Top	sum *ttd*	54,957.87	325,264.91	408,944.34	334,137.46	455,029.66	239,364.16	263,777.37	286,025.59	789,167.12
Overhanging	avg. *time* [s]	2,930.55	3,557.69	3,597.05	3,362.72	3,533.29	3,405.64	3,388.80	3,549.58	3,448.01
P5 –	sum v_{used}	547.20	3,651.20	4,204.00	3,166.80	5,235.60	2,719.20	2,805.60	2,877.60	8,402.40
LBS	sum *ttd*	55,039.11	299,473.10	376,064.06	307,130.34	423,445.93	236,839.91	243,617.19	250,119.17	730,576.27
Simple	avg. *time* [s]	2,720.65	2,705.81	2,864.45	2,366.26	3,178.21	2,627.47	2,816.95	2,872.28	2,772.23
P6 –	sum v_{used}	540.40	3,643.80	4,199.80	3,160.60	5,223.40	2,708.00	2,795.40	2,880.60	8,384.00
LBS	sum *ttd*	54,906.67	298,749.57	375,314.07	306,732.75	422,237.56	236,297.09	242,936.01	249,737.21	728,970.31
Complete	avg. *time* [s]	2,724.63	2,712.28	2,858.05	2,356.86	3,189.25	2,633.30	2,811.51	2,874.37	2,773.06
P7 –	sum v_{used}	515.20	3,716.60	4,227.80	3,193.20	5,266.40	2,617.80	2,831.60	3,010.20	8,459.60
Reachability	sum *ttd*	53,478.93	301,765.06	376,079.01	307,862.47	423,460.53	230,583.85	244,564.20	256,174.95	731,323.00
	avg. *time* [s]	2,182.89	2,146.41	2,445.83	1,867.57	2,679.38	1,849.18	2,393.33	2,577.91	2,273.47
P8 –	sum v_{used}	512.00	3,693.00	4,159.00	3,176.20	5,187.80	2,621.20	2,796.80	2,946.00	8,364.00
Axle Weights	sum *ttd*	53,158.60	301,120.47	370,416.22	307,143.14	417,552.15	230,521.79	242,202.78	251,970.72	724,695.29
	avg. *time* [s]	1,169.59	1,498.99	2,146.13	1,172.43	2,211.49	1,319.30	1,828.31	1,928.28	1,691.96

Table 15 (continued)

		m					*Item types*			Total
		20	60	100	200	400	3	10	100	
P9 –	sum v_{used}	512.00	3,698.20	4,160.20	3,183.80	5,186.60	2,623.20	2,793.00	2,954.20	8,370.40
Balanced	sum *ttd*	53,157.73	300,632.32	370,127.48	307,027.11	416,890.42	230,527.69	241,425.81	251,964.03	723,917.53
Load	avg. *time* [s]	1,162.88	1,515.99	2,174.62	1,199.33	2,218.32	1,294.69	1,872.34	1,959.43	1,708.82
P10 –	sum v_{used}	622.80	4,394.00	5,098.40	3,824.80	6,290.40	3,083.80	3,360.20	3,671.20	10,115.20
Combination	sum *ttd*	58,393.46	345,117.59	429,896.70	348,135.14	485,272.61	258,948.62	277,849.61	296,609.52	833,407.75
	avg. *time* [s]	3,138.68	3,547.99	3,596.32	3,422.74	3,548.19	3,446.21	3,464.22	3,545.96	3,485.46

Funding Open Access funding enabled and organized by Projekt DEAL.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

Baker B, Coffman E, Rivest R (1980) Orthogonal packings in two dimensions. SIAM J Comput 9(4):846–855. https://doi.org/10.1137/0209064

Bischoff EE (2003) Dealing with load bearing strength consideration in container loading problemseuropean business management school. University of Wales, Swansea

Bischoff EE, Ratcliff MSW (1995) Issues in the development of approaches to container loading. Omega 23(4):377–390. https://doi.org/10.1016/0305-0483(95)00015-G

Bortfeldt A (2012) A hybrid algorithm for the capacitated vehicle routing problem with three-dimensional loading constraints. Comput Oper Res 39(9):2248–2257. https://doi.org/10.1016/j.cor.2011.11.008

Ceschia S, Schaerf A, Stützle T (2013) Local search techniques for a routing-packing problem. Comput Ind Eng 66(4):1138–1149. https://doi.org/10.1016/j.cie.2013.07.025

Clarke G, Wright JW (1964) Scheduling of vehicles from a central depot to a number of delivery points. Oper Res 12(4):568–581, URL http://www.jstor.org/stable/167703

Fuellerer G, Doerner KF, Hartl RF, Iori M (2009) Ant colony optimization for the two-dimensional loading vehicle routing problem. Comput Oper Res 36(3):655–673. https://doi.org/10.1016/j.cor.2007.10.021

Fuellerer G, Doerner KF, Hartl RF, Iori M (2010) Metaheuristics for vehicle routing problems with three-dimensional loading constraints. Eur J Oper Res 201(3):751–759. https://doi.org/10.1016/j.ejor.2009.03.046

Gendreau M, Iori M, Laporte G, Martello S (2006) A Tabu Search Algorithm for a Routing and Container Loading Problem. Transp Sc 40(3):342–350. https://doi.org/10.1287/trsc.1050.0145

Junqueira L, Morabito R, Yamashita DS, Yanasse HH (2013) Optimization Models for the Three-Dimensional Container Loading Problem with Practical Constraints. Springer, NY, pp 271–293. https://doi.org/10.1007/978-1-4614-4469-5

Karabulut K, İnceoğlu MM (2005) A hybrid genetic algorithm for packing in 3d with deepest bottom left with fill method. In: Yakhno T (ed) Advances in Information Systems. Springer Berlin Heidelberg, Berlin, Heidelberg, pp 441–450

Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220(4598):671–80

Koch H, Bortfeldt A, Wäscher G (2018) A hybrid algorithm for the vehicle routing problem with backhauls, time windows and three-dimensional loading constraints. OR Spect 40(4):1029–1075. https://doi.org/10.1007/s00291-018-0506-6

Krebs C, Ehmke JF (2021) Axle weights in combined vehicle routing and container loading problems. EURO J Transp Logist 10:100043. https://doi.org/10.1016/j.ejtl.2021.100043

Lodi A, Martello S, Vigo D (1999) Heuristic and Metaheuristic Approaches for a Class of Two-Dimensional Bin Packing Problems. INFORMS J Comput 11(4):345–357. https://doi.org/10.1287/ijoc.11.4.345

Mak-Hau V, Moser I, Aleti A (2018) An exact algorithm for the heterogeneous fleet vehicle routing problem with time windows and three-dimensional loading constraints. In: Sarker R, Abbass HA, Dunstall S, Kilby P, Davis R, Young L (eds) Data and Decision Sciences in Action. Springer International Publishing, Cham, pp 91–101. https://doi.org/10.1007/978-3-319-55914-8

Moura A (2008) A Multi-Objective Genetic Algorithm for the Vehicle Routing with Time Windows and Loading Problem. Gabler, Wiesbaden, pp 187–201. https://doi.org/10.1007/978-3-8349-9777-7

Moura A, Oliveira JF (2009) An integrated approach to the vehicle routing and container loading problems. OR Spect 31(4):775–800. https://doi.org/10.1007/s00291-008-0129-4

Pace S, Turky A, Moser I, Aleti A (2015) Distributing fibre boards: A practical application of the heterogeneous fleet vehicle routing problem with time windows and three-dimensional loading constraints. Procedia Computer Science 51:2257–2266. https://doi.org/10.1016/j.procs.2015.05.382, URL http://www.sciencedirect.com/science/article/pii/S1877050915011904, international Conference On Computational Science, ICCS 2015

Ratcliff SMW, Bischoff EE (1998) Allowing for weight considerations in container loading. OR Spectrum 20:65–71. https://doi.org/10.1007/s002910050053

Reil S, Bortfeldt A, Mönch L (2018) Heuristics for vehicle routing problems with backhauls, time windows, and 3d loading constraints. European Journal of Operational Research 266(3):877–894. https://doi.org/10.1016/j.ejor.2017.10.029, URL http://www.sciencedirect.com/science/article/pii/S0377221717309426

Ropke S, Pisinger D (2006) A unified heuristic for a large class of vehicle routing problems with backhauls. European Journal of Operational Research 171(3):750–775, https://doi.org/10.1016/j.ejor.2004.09.004, URL http://www.sciencedirect.com/science/article/pii/S0377221704005831, feature Cluster: Heuristic and Stochastic Methods in Optimization Feature Cluster: New Opportunities for Operations Research

Solomon MM (1987) Algorithms for the vehicle routing and scheduling problems with time window constraints. Oper Res 35(2):254–265. https://doi.org/10.1287/opre.35.2.254

Tarantilis CD, Zachariadis EE, Kiranoudis CT (2009) A hybrid metaheuristic algorithm for the integrated vehicle routing and three-dimensional container-loading problem. IEEE Trans Intell Transp Syst 10(2):255–271. https://doi.org/10.1109/TITS.2009.2020187

Wei L, Zhang Z, Lim A (2014) An adaptive variable neighborhood search for a heterogeneous fleet vehicle routing problem with three-dimensional loading constraints. IEEE Comput Intell Magaz 9(4):18–30. https://doi.org/10.1109/MCI.2014.2350933

Zhang D, Cai S, Ye F, Si YW, Nguyen TT (2017) A hybrid algorithm for a vehicle routing problem with realistic constraints. Inf Sci 394–395:167–182. https://doi.org/10.1016/j.ins.2017.02.028

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Authors and Affiliations

Corinna Krebs[1] · Jan Fabian Ehmke[2] · Henriette Koch[1]

Jan Fabian Ehmke
Jan.Ehmke@univie.ac.at

Henriette Koch
Henriette.Koch@ovgu.de

[1] Department of Management Science, Otto von Guericke University Magdeburg, Universitätsplatz 2, 39106 Magdeburg, Germany

[2] Business Analytics Group, University of Vienna, Kolingasse 14-16, 1090 Vienna, Austria