OPTIMIZING STRONGLY RESTRICTED LOADING PROBLEMS WITH CONTAINERS AND PALLETS

DISSERTATION

CORINNA S. KREBS

Optimizing
strongly restricted loading problems
with containers and pallets

Schriftliche Promotionsleistung
zur Erlangung des akademischen Grades
Doctor rerum politicarum

vorgelegt und angenommen
an der Fakultät für Wirtschaftswissenschaft
der Otto-von-Guericke-Universität Magdeburg

Verfasserin: Corinna S. Krebs
Geburtsdatum und -ort: 06.07.1993, Bietigheim-Bissingen

Arbeit eingereicht am: 27.07.2023

Gutachter der schriftlichen Promotionsleistung:
Prof. Dr. Jan Fabian Ehmke; Prof. Dr. Marlin Ulmer

Datum der Disputation: 26.10.2023

Abstract

The entire logistics world is changing: the value chain, from raw material extraction to recycling, is experiencing a digital transformation that aims to increase data availability and simultaneously improve the connectivity of operational processes. This offers new opportunities for better and more realistic logistics planning that have yet to be investigated. This thesis concerns algorithms for strongly restricted loading problems in the context of the container loading problem combined with vehicle routing problems. Thus, new loading constraints are introduced as existing ones are improved – prompting the question of its costs and associated benefits. The costs are reflected by adapted objective functions. The benefits can be divided into three categories: 1) higher security (for drivers, road users, parcels, and vehicles), 2) efficient processes, as well as 3) adherence to legal requirements. The thesis comprises a total of six scientific works dealing with this topic.

Paper I[1] describes various implementations of the Deepest-Bottom-Left-Fill algorithm, a widely used algorithm to solve the container loading problem. It is combined with an adaptive large neighborhood search heuristic to solve the vehicle routing problem. The computational experiments evaluate the best implementation and the impact of various constraints on the objective function and runtime.

Paper II[2] introduces formulas for considering axle weights for different truck types, e.g., varying axle configuration and trailer presence. It uses the same algorithms as the first paper. The computational experiments show that including the axle weights constraints has only a minor impact on the objective function but positive effects on the runtime. Therefore, the axle weights should always be considered in models for security reasons.

Paper III[3] focuses on analyzing complex loading constraints, presenting weaknesses of current formulations for vertical stability and stacking constraints, and introducing new constraints. The largest constraint set, consisting of geometry, orthogonality, rotation, vertical stability, stacking, reachability, axle weights, and balanced load constraints, is analyzed in the computational experiments. These constraints are evaluated independently concerning their impact on the objective function and runtime. A central conclusion is that the combined constraints' impact is less than the sum of each constraint.

Paper IV[4] introduces a new formulation based on the science of statics for the vertical stability constraint, achieving the best results regarding impacts on the objective function and runtime compared to other common formulations.

Paper V[5] deals with calculating the manual effort for loading and unloading. This effort can be time-wise evaluated using the Methods-Time-Measurement (MTM) approach. This can also enable rearrangements of items so that the constraints dealing with fixed unloading sequences (e.g., LIFO) are replaced. The computational experiments show that this approach positively affects the objective values and the runtime.

Paper VI[6] introduces two tools for the combined optimization problem. The first one, "Solution

[1]Krebs, C. et al. (2023). "Effective loading in combined vehicle routing and container loading problems". In: *Computers & Operations Research* vol. 149, p. 105988. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2022.105988. URL: https://www.sciencedirect.com/science/article/pii/S0305054822002258

[2]Krebs, C. and Ehmke, J. F. (2021a). "Axle Weights in combined Vehicle Routing and Container Loading Problems". In: *EURO Journal on Transportation and Logistics* vol. 10, p. 100043. ISSN: 2192-4376. DOI: 10.1016/j.ejtl.2021.100043. URL: https://www.sciencedirect.com/science/article/pii/S2192437621000157

[3]Krebs, C. et al. (2021). "Advanced loading constraints for 3D vehicle routing problems". In: *OR Spectrum*. ISSN: 1436-6304. DOI: 10.1007/s00291-021-00645-w. URL: https://doi.org/10.1007/s00291-021-00645-w

[4]Krebs, C. and Ehmke, J. F. (2021b). "Vertical Stability Constraints in Combined Vehicle Routing and 3D Container Loading Problems". In: *Computational Logistics*. Ed. by Mes, M. et al. Cham: Springer International Publishing, pp. 442–455. ISBN: 978-3-030-87672-2

[5]Krebs, C. (2023). "Manual Unloading in 3D Loading Vehicle Routing Problems. Submitted to EURO Journal on Transportation and Logistics on 19.07.2023"

[6]Krebs, C. and Ehmke, J. F. (May 2023). "Solution validator and visualizer for (combined) vehicle routing and container loading problems". In: *Annals of Operations Research*. ISSN: 1572-9338. DOI: 10.1007/s10479-023-05238-0. URL: https://doi.org/10.1007/s10479-023-05238-0

Validator", can check solutions concerning modifiable constraint sets, whereby all constraints evaluated in the previous papers are implemented. The second tool includes this Solution Validator and can visualize solutions for the routes, including distances and travel times. In addition, the container loading spaces, including the position of all boxes, are presented.

Overall, each paper represents a single piece from a broader puzzle, highlighting constraints from different perspectives.

Contents

List of Papers

Paper I

Krebs, C. et al. (2023). "Effective loading in combined vehicle routing and container loading problems". In: *Computers & Operations Research* vol. 149, p. 105988. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2022.105988. URL: https://www.sciencedirect.com/science/article/pii/S0305054822002258

Paper II

Krebs, C. and Ehmke, J. F. (2021a). "Axle Weights in combined Vehicle Routing and Container Loading Problems". In: *EURO Journal on Transportation and Logistics* vol. 10, p. 100043. ISSN: 2192-4376. DOI: 10.1016/j.ejtl.2021.100043. URL: https://www.sciencedirect.com/science/article/pii/S2192437621000157

Paper III

Krebs, C. et al. (2021a). "Advanced loading constraints for 3D vehicle routing problems". In: *OR Spectrum*. ISSN: 1436-6304. DOI: 10.1007/s00291-021-00645-w. URL: https://doi.org/10.1007/s00291-021-00645-w

Paper IV

Krebs, C. and Ehmke, J. F. (2021b). "Vertical Stability Constraints in Combined Vehicle Routing and 3D Container Loading Problems". In: *Computational Logistics*. Ed. by Mes, M. et al. Cham: Springer International Publishing, pp. 442–455. ISBN: 978-3-030-87672-2

Paper V

Krebs, C. "Manual Unloading in 3D Loading Vehicle Routing Problems". *Submitted to* EURO Journal on Transportation and Logistics *on 19.07.2023*

Paper VI

Krebs, C. and Ehmke, J. F. (May 2023). "Solution validator and visualizer for (combined) vehicle routing and container loading problems". In: *Annals of Operations Research*. ISSN: 1572-9338. DOI: 10.1007/s10479-023-05238-0. URL: https://doi.org/10.1007/s10479-023-05238-0

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

The global logistic sector is expected to grow significantly in the coming years: from 5.99 billion euros in 2021 to 6.88 billion euros in just three years (2024)[1], representing an increase of 14%. This is particularly evident in the Courier, Express and Parcel (CEP) industry. In less than a decade, the global parcel shipping volume has quadrupled, from 43 billion parcels sent in 2014 to over 159 billion in 2021. However, this is only the beginning of a rapidly growing, globally impactful scale. In the next few years, the volume is expected to reach between 216 billion and 300 billion parcels by 2027[2].

To meet these challenging future forecasts and to remain competitive, unprecedented investments into the digital transformation of the supply chain management are necessary: A poll by McKinsey & Company, Inc. 2021, for example, reveals that most surveyed companies have invested in technologies to digitize the supply chain, independently of the industry sector. Visibility and specific planning tools are thus the major factors for investments. Furthermore, a survey conducted by SCI Verkehr GmbH 2022 with 200 companies in the German logistics industry on planned investments for 2021 confirms that investments will primarily flow into logistics software (69% of all participants). Through these investments in digital transformation, more and more data concerning parcels and vehicles becomes available. Accordingly, this data should be used to further improve the logistical planning.

However, the question arises to what degree the additional data influences the costs compared to the received benefits. To be more concrete, realistic planning leads to three effects (benefits): 1) increased security, 2) adherence to legal requirements, and 3) more efficient operations. Higher security (1) is ensured for the driver, road users, parcels and the vehicle. For example, vertical stability constraints ensure stable packing and prevent items from falling onto other items or the driver. In addition, balanced loads should be considered to avert that goods move inside the loading space leading to vehicles tipping over and causing accidents. Legal requirements (2) can be obeyed through the availability of data, e.g., enabling the calculation and adherence of the vehicle's axle weights directly in the planning phase. 3) More efficient operations are achieved by, e.g., ensuring that parcels can be loaded or unloaded without obstructions leading to optimal process times.

In addition to the benefit side already shown before, there is the cost side represented by the impact on the objective values (e.g., total travel distance and the number of used vehicles) leading to higher costs, for example in terms of fuel consumption, driver costs, and fixed costs for additional vehicles. Another aspect is the runtime's influence depending on the algorithms' complexity and constraints. The evaluation of the cost-benefit ratio for different settings is the main objective of this thesis.

1.1 Optimization Problem

This thesis focuses on loading constraints in terms of new formulations, new approaches, and fixing existing corner cases. It thus presents the costs and benefits of various loading constraints. This is evaluated in the context of the vehicle routing problem, e.g., the impact on the total travel distance. Therefore, the combined optimization problem consisting of the vehicle routing problem and the 3D container loading problem (3L-CVRP) introduced by Gendreau et al. 2006 and the extension with time windows (3L-VRPTW) are investigated. These optimization problems consist in delivering 3D parcels (items) from a central depot to a set of customers by a fleet of homogeneous vehicles while obeying routing and loading constraints. This topic is highly relevant as it mirrors typical applications in practice, such as the parcel delivery industry or the delivery of groceries and furniture. Typically, the following five loading constraints are considered: Geometry, Orthogonality, Rotation, Last-In-First-Out (LIFO), Fragility, and Minimal Supporting Area. The Geometry constraint ensures adherence to the dimensions of items and vehicles and prevents the overlapping

[1] see Transport Intelligence Ltd 2021

[2] see Pitney Bowes Inc. 2023

of objects. The Orthogonality constraint requires that the edges of the items are parallel to the vehicle walls. The rotation of items is only allowed along the length-width plane. LIFO means that the unloading of items must be done by straight movements, parallel to the vehicle walls, and without rearrangements or blocked by any obstacles. The Fragility constraint distributes items into fragile and non-fragile ones so that non-fragile items can be placed only on top of other non-fragile items. The Minimal Supporting Area constraint requires a specific supporting area for every item provided by underlying items or the vehicle floor in relation to the basic area. Although dozens of papers have investigated the 3L-CVRP and the 3L-VRPTW in the past 15 years, the number dealing with new or improved loading constraints amounts to a mere handful. Therefore, this dissertation fills this research gap by highlighting loading constraints.
As described above, several new loading constraints are introduced, or, in the case of corner cases, existing formulations are improved. For example, new loading constraints ensuring vertical stability are formulated as the current constraint in this field, the Minimal Supporting Area, which can lead to unstable item stacks. Another example is the Fragility constraint which, so far, only classifies items into fragile and non-fragile items. For more realistic modeling, the new Load Bearing Strength constraint for considering the actual load acting on an item is introduced based on the science of statics. In addition, constraints to ensure a balanced load along the vehicle halves are also evaluated. Moreover, formulas for calculating axle weights for different vehicle types, trailers, and configurations are introduced. In the case of manual loading and unloading, constraints obeying the reachability of items and the possibility to lift items while handling (adaption of LIFO) are analyzed. Another focus is examining the unloading effort for items and the possibility of rearranging items that lead to an infeasible placement according to the LIFO constraint. Finally, two tools are presented for validating and visualizing solutions and analyzing adherence to the loading constraints.

1.2 Thesis outline

This thesis comprises six scientific papers in three sections, highlighting loading constraints from different perspectives. The relationship between these papers is presented in Fig. 1.1. All papers deal with the combined vehicle routing and container loading problem (3L-CVRP and/or 3L-VRPTW).

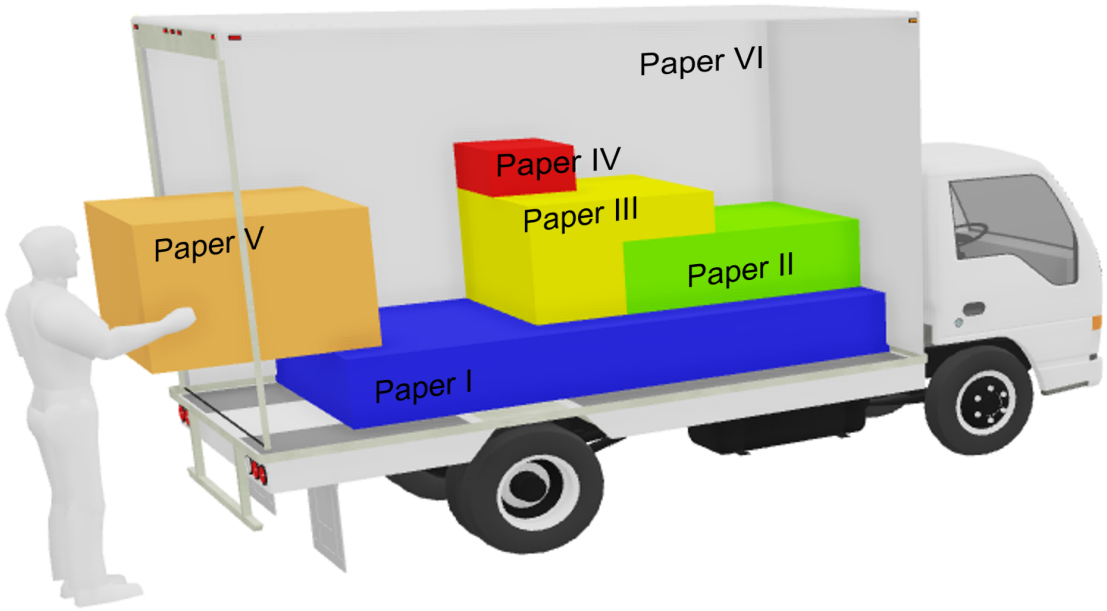


Figure 1.1: Relation between Papers

Paper I builds the basis of further four papers. The paper focuses on two aspects: First, it presents and evaluates three implementation variants of the Deepest-Bottom-Left-Fill algorithm initially invented by Karabulut and İnceoğlu 2005, which differ in the representation of available placement positions and the sliding technique of items. As a result, the Deepest-Bottom-Left-Fill algorithm with Spaces is the best compared to the other variants; therefore, it is also used in Papers II-V. In addition, the paper shows that every loading constraint has a different degree of influence on the objective values and runtime. Consequently, further research is relevant and worthwhile.

Paper II introduces simple yet adjustable formulas for considering the Axle Weight constraint based on the science of statics. Considering axle weights is crucial, as overloaded axles lead to higher braking distance and more severe accidents. The introduced formulas can be adapted to various truck types that differ in axle configurations and the presence of trailers. For the computational experiments, the algorithm introduced in Paper I is used. The evaluation focuses on the impact of including axle weights in the combined problem. Interestingly, the axle weight constraint has only small negative impacts on the objective values but clearly decreases the runtime. Combined with the fact of increased security, it should always be included in the models.

Paper III enhances the combined problem by several complex real-world loading constraints, which lead to increased security, such as the reachability of items, the balanced load within the vehicle loading space, and the distribution of items' masses within a stack. Additionally, it presents new definitions for vertical stability. Consequently, this paper currently considers the largest constraint set. A new instance set is created to better evaluate the loading constraints, varying in the number of customers, item types, and items. For instance sets from the literature, additional parameters necessary for the loading constraints are added. All constraints are evaluated concerning their impact on the objective values.

Paper IV demonstrates differences and corner cases of the well-known vertical stability formulations in literature. Based on this and the science of statics, a new variant of the vertical stability constraint is introduced. In the computational studies, the new formulation is compared with various vertical stability constraints from the literature and those introduced in Paper III. It performs best with regard to objective values and has a comparable runtime.

Paper V includes the effort for unloading an item in the objective function based on the well-known Methods-Time Measurement (MTM) analysis. Moreover, in contrast to the previous approaches, it accepts infeasible item positions according to the LIFO. For these infeasible positions, the necessary reloading effort is calculated. This time-wise additional effort is also added to the travel time. The paper shows that this approach is highly effective concerning performance and runtime.

Paper VI introduces two open-source tools to support further research in this field and to increase the transparency of solutions: One for the validation of the feasibility of solutions ("Solution Validator") and one for the visualization of solutions ("Visualizer"), which also integrates the validation tool to highlight violated routing or loading constraints. Moreover, this paper summarizes the currently best-known and validated solutions for various instance sets.

References

Gendreau, M., Iori, M., Laporte, G., and Martello, S. (2006). "A Tabu Search Algorithm for a Routing and Container Loading Problem". In: *Transportation Science* vol. 40, no. 3, pp. 342–350. ISSN: 0041-1655. DOI: 10.1287/trsc.1050.0145. URL: http://pubsmisc.informs.org/doi/abs/10.1287/trsc.1050.0145.

Karabulut, K. and İnceoğlu, M. M. (2005). "A Hybrid Genetic Algorithm for Packing in 3D with Deepest Bottom Left with Fill Method". In: *Advances in Information Systems*. Ed. by Yakhno, T. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 441–450. ISBN: 978-3-540-30198-1.

McKinsey & Company, Inc. (Nov. 2021). *How COVID-19 is reshaping supply chains*. URL: https://www.mckinsey.com/business-functions/operations/our-insights/how-covid-19-is-reshaping-supply-chains.

Pitney Bowes Inc. (Mar. 2023). *Parcel Shipping Index 2022*. URL: https://www.pitneybowes.com/content/dam/pitneybowes/us/en/shipping-index/22-pbcs-04529-2021-global-parcel-shipping-index-ebook-web-002.pdf.

SCI Verkehr GmbH (Jan. 2022). *Das SCI/Logistikbarometer*. URL: https://www.sci.de/fileadmin/user_upload/logistikbarometer/pdf/SCI_LOGISTIKBAROMETER_Jan_2022.pdf.

Transport Intelligence Ltd (Jan. 2021). *Total Logistics 2021*. URL: https://www.ti-insight.com/press_releases/global-logistics-market-forecast-to-grow-by-a-cagr-of-4-7-to-2024.

Papers

Paper I

Effective Loading in Combined Vehicle Routing and Container Loading Problems

Corinna Krebs, Jan Fabian Ehmke, Henriette Koch

Published in *Computers & Operations Research*, Volume 149, January 2023, 105988

Abstract

This paper addresses more effective loading within the 3L-VRPTW, which is a combination of the vehicle routing problem with time windows and 3D loading constraints. We use a hybrid algorithm consisting of an outer Adaptive Large Neighborhood Search tackling the routing problem in combination with an inner Deepest-Bottom-Left-Fill algorithm solving the container loading problem. We propose and compare three new variants for the Deepest-Bottom-Left-Fill algorithm which differ in the representation and storage of available possible placement positions and the shift of items. The possible placement positions can be determined either by available points so that (1) a non-overlapping check between the items is necessary, (2) by available free spaces, or (3) by using a Rectangle Tree (RTree), where items and their positions are stored in a tree. For computational studies, two well-known instance sets are used. The algorithms are evaluated and compared concerning their solution quality and performance. Hereby, the algorithm with free spaces receives the best results with the smallest runtime. Moreover, the impact of different loading constraints is analyzed showing that the LIFO and minimal supporting area constraints have significant effects on the total travel distance. Several new best solutions were found. All results are validated and published at GitHub.

Contents

1 Introduction

The efficient usage of loading space of containers is of great economic importance, especially in the area of container ship loading, pallet loading, warehouse management, and aircraft freight management. One approach solving this complex problem is the Deepest-Bottom-Left-Fill (DBLF) algorithm introduced by Karabulut and İnceoğlu (2005). The DBLF algorithm is usually embedded in the optimization of superior problems like container loading problems or the vehicle routing problem (namely 3L-CVRP). It has even been adapted to the field of additive manufacturing (Araújo et al. 2019). The advantages of the DBLF-algorithm are clear: it is simple and has a good trade-off between runtime and solution quality. However, when the number of items becomes large, DBLF shows its weakness: Whenever a current item is to be placed, a non-overlapping check with all already placed items inside the loading space has to be carried out. This might need to be repeated for several possible positions until a feasible position for the current item is found. Consequently, the original non-overlapping check proposed by Karabulut and İnceoğlu (2005) has a high computational complexity.

Being a subproblem of a superior optimization problem, the DBLF algorithm has to be implemented as efficiently as possible. When combined with the vehicle routing problem with time-windows (3L-VRPTW), Koch et al. (2019) showed that a combination of an Adaptive Large Neighborhood Search (ALNS) for the routing problem and the DBLF algorithm for the load optimization is competitive. We build on this solution framework in this paper and investigate the performance of three new DBLF implementation variants in the context of the 3L-VRPTW. We do so by analyzing a large number of different loading constraints sets. In particular, we investigate efficient ways of the non-overlapping check (I); we evaluate the DBLF algorithm solely and in conjunction with the combined container loading and vehicle routing problem with time windows (3L-VRPTW) (II), and we analyze the impact of each loading constraint on the objective value (III) and we provide new best-solutions for two well-known instance sets (IV).

Concerning (I), we implement three algorithms: one is inspired by current industry practice; one is based on academic literature, namely Karabulut and İnceoğlu (2005); and one is a new variant where a non-overlapping check is not necessary. In the first algorithm, coming from the industrial environment, a Rectangle Tree (RTree) is used. RTree structures are common to evaluate spatial data quickly, e.g. for data analysis or collision detection in the field of three-dimensional computer graphics and game development. In an RTree, the objects and their positions are stored in a tree structure enabling a fast check for non-overlapping of the current item with all already placed items. The idea of using an RTree in the context of the DBLF algorithm was first proposed by Allen et al. (2011) and is further developed in this paper by exploring more loading constraints. The second one is based on Koch et al. (2019) and uses the non-overlapping check as proposed in Karabulut and İnceoğlu (2005). In the last approach, free available spaces are determined so that a non-overlapping check is not necessary to place items.

Concerning (II), we evaluate the three DBLF algorithm variants in the context of the 3L-VRPTW. While in classical packing problems packing algorithms focus on the maximization of the volume, combining packing and vehicle routing requires the quick provision of good solutions for a wide variety of routes and customer sequences. Hence, our DBLF algorithm variants need to be evaluated regarding effective interaction with routing algorithms, esp. with handling a large number of routes.

Concerning (III), we will present the results of three computational studies. In the first one, the DBLF algorithm variants are called with predefined routes to enable comparison with the same computing resources. In the second study, the combined problem is solved by a hybrid algorithm

as shown by Koch et al. (2018). The third study deals with a sensitivity analysis of the loading constraints. For the second and the third study, an outer ALNS determines a set of routes. Then, for this set of routes, one of the DBLF algorithm variants is called to create a feasible packing plan for each route. The computational tests are conducted based on two well-known instance sets from the literature. We compare each variant concerning the solution quality (total travel distance), the runtime, and, in the case of the second study, also in the number of iterations. In the third study, we analyze the impact of the loading constraints on the total travel distance, e.g. the LIFO policy, the load capacity of vehicles, the consideration of support and fragility of items. To summarize, this paper presents improving factors for the DBLF-algorithm and the impact of different loading constraints.
The paper is organized as follows. In Section 2, the relevant literature is reviewed. The 3L-VRPTW is formulated in Section 3. In Section 4, the hybrid algorithm and the three DBLF algorithm variants are described. Section 5 presents computational experiments. Finally, conclusions are drawn in Section 6.

2 Literature Review

This paper presents three variants of the DBLF algorithm. These are evaluated in the context of the 3L-VRPTW, which is a combination of the vehicle routing problem with time windows (VRPTW) and 3D loading constraints. First, relevant literature for the DBLF algorithms is reviewed. Then, research on the related 3L-VRPTW is discussed.

2.1 Deepest-Bottom-Left-Fill Algorithms

The DBLF algorithm has its origin in Baker et al. (1980), who developed the Bottom-Left algorithm in the context of the 2D-strip packing problem. In this problem, two-dimensional items are packed into a container where one dimension is unknown and must be minimized. Their idea is to place items first into the lowest possible position and then move the items as left as possible along their vertical position. Hopper and Turton (2001) use this approach to solve the 2D container loading problem. To avoid large empty spaces, they modify the algorithm so that available holes are filled (Bottom-Left-Fill). To cover 3D container loading problems, Karabulut and İnceoğlu (2005) extend the Bottom-Left-Fill method and introduce the DBLF algorithm, which places items first in the deepest position. As in Karabulut and İnceoğlu (2005), the DBLF principle has often been implemented with genetic algorithms to solve 3D container loading problems (e.g. in Kang et al. 2012, Moon and Nguyen 2013, Feng et al. 2015 and Jamrus and Chien 2016) or implemented to solve 3D strip packing problems (e.g. Allen et al. 2011 and Wauters et al. 2013). The DBLF principle has not only been used for 3D container loading problems, but also in combination with vehicle routing problems (e.g. Ma et al. 2011, Zhu et al. 2012, Wu et al. 2013, Koch et al. 2018, Koch et al. 2019, and Krebs et al. 2021). In this problem field, the objective is to minimize the total travel distance so that the DBLF algorithms need to provide good solutions quickly rather than maximizing the volume, for example. Concerning best practices in container loading problems, we recommend Silva et al. (2019). In Araújo et al. (2019), the practicality of the DBLF principle is even analyzed for the area of additive manufacturing.
The DBLF principle has been extended by several variants, e.g. by adding further loading constraints. In Kang et al. (2012), the DBLF principle is embedded into a genetic algorithm and used in the context of the 3D Bin Packing Problem. Hereby, the items are allowed to be rotated in all ways. Moon and Nguyen (2013) tackle the 3D Bin Packing Problem by combining the DBLF principle with a greedy heuristic and a genetic algorithm. Additionally, they ensure a balanced loading w.r.t. the x- and y-axis of the loading space. Krebs and Ehmke (2021a) deal with the modeling of axle weights for different vehicle types. Hereby, the DBLF principle is used with an ALNS for solving the combined vehicle routing and container loading problem. The same algorithm is applied in Krebs et al. (2021) for a comprehensive study of new loading constraints (e.g. the reachability of items and realistic load distribution of stacked items). This algorithm is also used in Krebs and Ehmke (2021b), who focus on vertical stability constraints and introduce a new approach enabling stable packing. To extend the research of the impacts of loading constraints on objective values, we evaluate this aspect for the basic loading constraints as introduced by Gendreau et al.

(2006) in this paper.
Another variant of the DBLF algorithm considers its complexity: In the implementation by Karabulut and İnceoğlu (2005), the non-overlapping of items is ensured by checking the current item with all already placed items. To reduce the complexity of the algorithm, Allen et al. (2011) store the positions of the placed items in an RTree enabling an efficient non-overlapping check of items. Their algorithm achieved several new best-known solutions. Therefore, the potential of RTree representations is further explored in this paper (see “DBLF with RTree” algorithm) in the context of more loading constraints.

2.2 3L-VRPTW

Gendreau et al. (2006) introduce the combination of the vehicle routing and 3D container loading problem (namely 3L-CVRP). This problem involves the optimal planning of routes to deliver goods to customers that are located in a depot. This is accomplished by a fleet of vehicles having a certain load capacity. Gendreau et al. (2006) solve the vehicle routing problem (VRP) with an “outer" tabu search, which determines the routes. The loading problem is tackled by an iteratively invoked “inner" tabu search. The 3L-VRPTW is a problem variant of the 3L-CVRP, where the depot and the customer locations have time windows. In Pace et al. (2015), a heuristic based on simulated annealing and an iterated local search is proposed to solve the VRPTW. Since they examine the distribution of fiber boards, a specialized loading heuristic and a balanced loading constraint are necessary. The latter is also adopted by Mak-Hau et al. (2018), who develop a mixed-integer linear program for a simplified version of the 3L-VRPTW and a heterogeneous fleet. Zhang et al. (2017) solve the 3L-VRPTW with a hybrid approach where the routing heuristic is based on a tabu search and an artificial bee colony algorithm. The loading heuristic is a combination of a personification heuristic and simulated annealing. They combine two well-known instance sets provided by Gendreau et al. (2006) and Solomon (1987).
As the combination of (meta-)heuristics has proven to be an efficient way to deal with the 3L-VRPTW, in this paper, we use the approach by Krebs et al. (2021) to test the effectiveness of different DBLF algorithm variants. In this hybrid algorithm, an outer ALNS combined with Simulated Annealing solves the routing problem, while an inner DBLF algorithm generates a packing plan for the created set of routes.

3 Problem Description

In the following, we present a problem description for the 3L-VRPTW, which we use to evaluate the different DBLF algorithm variants proposed in this paper. A complete mathematical formulation is provided in A. Adapting the convention by Koch et al. (2018), the 3L-VRPTW is specified 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 defined thanks to the following three times at 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, homogeneous 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 $(v = 1, ..., v_{used})$ represents 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) The number of used vehicles v_{used} does not exceed the number of available vehicles v_{max};

(S3) Each customer is visited exactly once;

(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:

(C1) Each route starts and terminates at the depot and visits at least one customer;

(C2) The vehicle does not arrive after the due date DD_i of any location i.

Each packing plan PP_v of a route R_v must obey the following loading constraints:

(L1) *Geometry*: The items must be packed within the vehicle ensuring non-overlapping;

(L2) *Orthogonality*: The items can only be placed orthogonally inside a vehicle;

(L3) *Rotation*: The items can be rotated 90° only on the width-length plane;

(L4) *Load Capacity*: The sum of masses of all included items of a vehicle does not exceed the maximum load capacity D;

(L5) *LIFO*: No item is placed above or in front of item $I_{i,k}$, which belongs to a customer served after customer i;

(L6) *Minimal Supporting Area*: Each item has a supporting area of at least a percentage α of its base area;

(L7) *Fragility*: No non-fragile items are placed on top of fragile items.

The 3L-VRPTW aims at determining a feasible solution minimizing the total travel distance *ttd* and meeting all above constraints.

4 Hybrid Solution Approach

To evaluate our DBLF algorithm variants in the context of the 3L-VRPTW, we employ a hybrid solution approach consisting of a routing algorithm for creating feasible routes and an embedded container loading algorithm, which generates feasible packing plans PP_v for the generated set of routes R_v. As a routing algorithm, we use the ALNS proposed by Koch et al. (2018). For the packing algorithm, one of the three investigated DBLF algorithm variants is used, which are presented in this section. We reference the line numbers of the algorithms in square brackets. Generally, we consider a solution to be feasible if all loading and routing constraints are obeyed.

4.1 Routing Algorithm

As a routing algorithm, we employ the ALNS proposed by Koch et al. (2018) adapted to the 3L-VRPTW without backhauls. The algorithm is shown in Alg. I.1 and briefly described in the following. For further details, we refer to the original paper.
Initially, a set of routes s_{init} is constructed [1] by the Savings Heuristic developed by Clarke and Wright (1964). Hereby, routing and loading constraints must be ensured. Consequently, the DBLF algorithm is called to check the loading feasibility.
The next feasible solution is determined by removing a randomized number of customers from the routes using removal heuristics [5-6]. The removed customers are reinserted using insertion heuristics [7-8]. The removal and insertion heuristics are described in detail in Koch et al. (2018).

Algorithm I.1 Adaptive Large Neighbourhood Search

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

construct initial feasible solution s_{init} using Savings Heuristic
$s_{best} := s_{init}$
$s_{curr} := s_{init}$
do
    select removal operator rem
    select number of customers to be removed n_{rem}
    select insertion operator $inst$
    determine next feasible solution $s_{next} := inst(rem(s_{curr}, n_{curr}))$
    for each route R_v **in** s_{next} **do**
        feasible := true
        if Deepest-Bottom-Left-Fill(R_v) not feasible **then**
            feasible := false
            break
        end if
    end for
    check acceptance of s_{next} using Simulated Annealing
    if feasible AND 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 it_p reached **then**
        update selection probabilities for insertion and removal heuristics
    end if
while one stopping criterion is not met

The new set of routes is checked for whether it fulfills all routing constraints, and whether a feasible packing plan can be created with the DBLF algorithm [9-15].
Infeasible solutions are always discarded. A feasible and superior solution (i.e. less total travel distance) is always accepted as the current solution. An inferior feasible solution may be accepted depending on the result of a Simulated Annealing Heuristic [16-17] developed by Kirkpatrick et al. (1983). Hereby, a worse but feasible solution is accepted with the probability $e^{-}(f*(s_{next} - f*(s))/T)$. T represents the current temperature ($T > 0$). The starting temperature is determined as proposed by Ropke and Pisinger (2006), i.e. a $w\%$ worse solution compared to the initial solution would be accepted with a probability of 0.5. After each iteration, the new temperature is calculated by multiplying the current temperature T by a cooling rate γ.
The current solution s_{curr} is then used to generate the next solution [18]. The success of the removal and insertion heuristics is evaluated after a defined number of iterations it_p to update the selection probabilities of these heuristics [23-25].
The algorithm stops if a time limit is reached or a total number of iterations is executed or the solution does not improve within a defined number of iterations [26].

4.2 Packing Algorithm

In this section, three DBLF algorithm variants are presented, which are based on the DBLF algorithm proposed by Karabulut and İnceoğlu (2005). 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.
For all algorithms, we assume that the point of origin of a Cartesian coordinate system is located in the deepest, bottommost, 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-axis. The position 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. First, 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 loading sequence IS reversed to the customer's visiting order. This sequence is used as an input parameter for the DBLF algorithm. Since each customer is visited exactly once (S3), all items of a customer must be placed in the loading space. If no feasible position for an item can be found, the route is revised, and a new one must be searched by the ALNS.

4.2.1 Deepest-Bottom-Left-Fill with Points

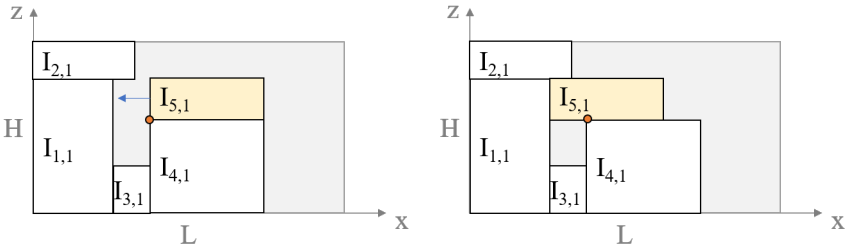
The following DBLF algorithm variant is the same as in Koch et al. (2019). We formalize this variant in Alg. I.2. The algorithm is rather close to the original proposed by Karabulut and İnceoğlu (2005) with the difference that the rotation of items is allowed here. In the set SP, all possible placement points for an item are stored. The first point in the set is the origin of the loading space [2]. Then, for each item I_p of the packing sequence IS and for each allowed orientation, a feasible position is searched [3-5]. For this purpose, the points of SP are successively tested as a possible position for the current item I_p. Based on the position, the item I_p is tried to slide further to the back, bottom, and left (see Fig I.1).

Algorithm I.2 Deepest-Bottom-Left-Fill with Points

Input: Instance data, smallest dimensions l_{min}, h_{min}
Output: Packing Plan PP_v, feasibility of R_v

```
1: initialize sorted sequence of unpacked items IS
2: initialize set of unique available points SP
3: for each item I_p ∈ IS do
4:     for each point po ∈ SP do
5:         for each permitted rotation do
6:             for each placed item I_q ∈ IS do
7:                 if I_p and I_q overlap then
8:                     continue with next orientation or point po
9:                 end if
10:            end for
11:            if position is feasible w.r.t. all loading constraints then
12:                save placement for I_p
13:                erase point po
14:                create and include new points in SP
15:                sort SP based on DBL
16:                for each point pi ∈ SP do
17:                    if pi and I_p overlap then
18:                        erase point pi
19:                    end if
20:                end for
21:                break
22:            end if
23:        end for
24:    end for
25:    if no feasible position found then
26:        return false
27:    end if
28: end for
```

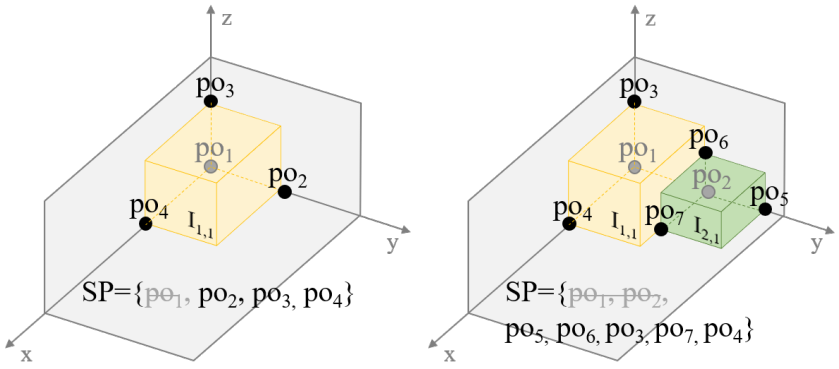
After sliding, for this current position for I_p, it is checked whether there is an intersection with any already placed item [7-10]. If this is not the case and thus the Geometry constraint (L1) is met, the position for item I_p is checked whether fulfilling the other loading constraints



(a) Point *po* for item $I_{1,5}$ (b) Sliding item to new position

Figure I.1: Sliding item $I_{5,1}$ based on current position

[11]. If the position is feasible, the position is saved for the item I_p [12], and the used point *po* is removed from SP [13]. Then, three new points are created as new possible placement points [14]: the bottom-right-back, the top-left-back, and the bottom-left-front points of the placed item (see Fig. I.2).



(a) After first placed item $I_{1,1}$ (b) After second placed item $I_{2,1}$

Figure I.2: Elements of set SP

Only points are included in SP providing that any item of the instance could be placed there. Thus, the shortest length or width l_{min} and smallest height h_{min} of any item of the instance are searched once and the points are checked whether their distances to the loading space walls are sufficient. After inserting the new points, the entire set SP is sorted based on the DBL policy [15]. In the last step, all possible points of SP are tested concerning whether they are covered by the placed item I_p. In this case, the points are removed from the SP [16-20].

4.2.2 Deepest-Bottom-Left-Fill with Spaces

As the previous approach shows, each item must be checked for non-overlapping with all already placed items. If there is an overlap, the next position for the current item is selected and the non-overlapping check is executed again. Due to this high complexity, the idea of the DBLF with Spaces was born. In this approach shown in Alg. I.3, the possible placement positions are stored as available free spaces instead of possible points. Thus, the effort for checking the non-overlapping between items and the vehicle is reduced.
Let S be the set of unique cuboids representing available free spaces for placing items. The general procedure is comparable to the previous one: The set SP is initialized with the entire loading space [2], sorted by the DBL-rule [11], and for each item I_p of the sequence IS, a possible position must be found [3]. Thus, each space *sp* of the set and each allowed rotation are tested as possible item positions until a feasible position is found, obeying all loading constraints [4-5]. In contrast to

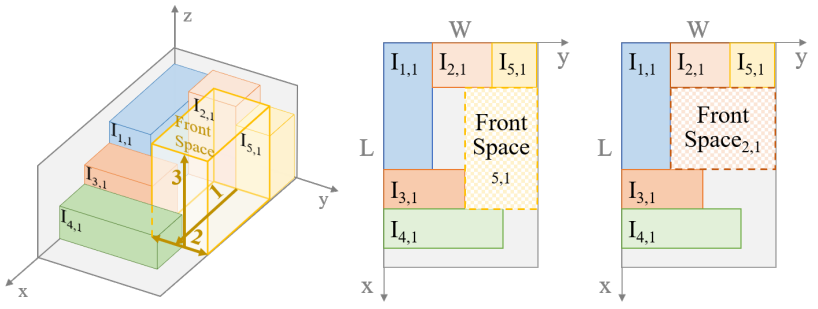
DBLF with Points, the items are not further slid. Moreover, instead of three new points, up to six spaces are created based on the current position of the placed item I_p.

Algorithm I.3 Deepest-Bottom-Left-Fill with Spaces

Input: Instance data
Output: Packing Plan PP_v, feasibility of R_v

```
initialize sorted sequence of items IS
initialize set of unique available spaces SP
for each item I_p ∈ IS do
    for each space sp ∈ SP do
        for each permitted rotation do
            if item I_p fits in space sp then
                if position is feasible w.r.t. all loading constraints then
                    save placement for I_p
                    erase space sp
                    create and include new spaces
                    sort SP based on DBL
                    get l_min and h_min of unplaced items ∈ IS
                    for each space si ∈ SP do
                        update space si
                        if si too small then
                            erase space si
                        end if
                        break
                    end for
                end if
            end if
        end for
    end for
    if no feasible position found then
        return false
    end if
end for
```

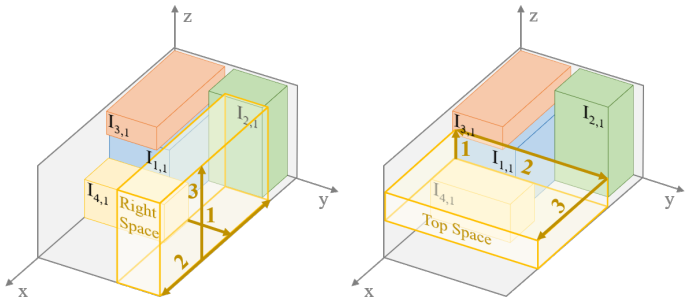
The first space is created by the front edge (*Front Space*) of item I_p. Starting from the front edge (minimum x-value), the maximum x-value is determined by expanding the space along the x-axis until reaching the vehicle door or another item. In the next step, the search for the minimum and maximum z-values is carried out, which are determined by the vehicle (floor or ceiling) or other items (underlying or overhanging ones). Then, the minimum and maximum y-values are searched by extending to the vehicle wall or to an item (see Fig. I.3a,b). The order in which the dimensions of the space are determined is decisive for the resulting space. For example, if the *Front Space* would expand along the y-axis first, the *Front Space* in Fig. I.3c occurs which corresponds to the *Front Space* of $I_{2,1}$ applying the current definition.



(a) Front Space for $I_{5,1}$ (b) Front Space for $I_{5,1}$ (c) Front Space for $I_{2,1}$

Figure I.3: Front Space Creation

The minimum y-value for the *Right Space* is defined by the item's right edge (see Fig. I.4a). Then, the maximum y-value is searched, which is defined either by an item or the vehicle wall. In the next step, the minimum and maximum values for the z-axis and then for the x-axis are searched. For the *Top Space*, the top edge determines the minimum z-value. Then, the minimum and maximum values along the y-axis and x-axis are searched (see Fig. I.4b). In addition, further three spaces are created if they are unique: (1) Another Front and (2) Right Space, where the minimum z-value represents the bottom edge of item I_p; (3) another Top Space, where the minimum x-value is the deepest edge of item I_p.



(a) Right Space for $I_{4,1}$ (b) Top Space for $I_{4,1}$

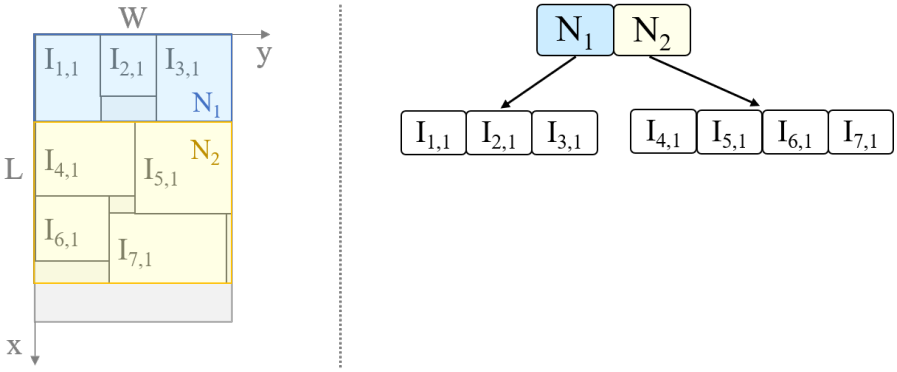
Figure I.4: Right and Top Space Creation

After the feasible placement of an item I_p, the used space sp is removed from the set [9], and the new spaces are inserted in the set SP [10]. To ensure that the available spaces of the set S represent empty volumes, the dimensions of all spaces are checked w.r.t. intersection with item I_p. If one or more spaces intersect with item I_p, then, these spaces are decreased so that no intersection occurs [14]. 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 (L1)). In contrast to the approach by Karabulut and İnceoğlu (2005) and the previous one, a non-overlapping check between each item is not necessary, which decreases the complexity. As in the previous approach, the possible placements must not exceed minimum dimensions. In contrast to the previous approach, the shortest length or width and height are not determined by the smallest items of the instance but rather by the smallest dimensions of unplaced items of the route. Therefore, the shortest length or width l_{min} and height h_{min} of any unplaced item of the route are searched [12], and all spaces which have smaller dimensions than the minimum one are removed [15-17].

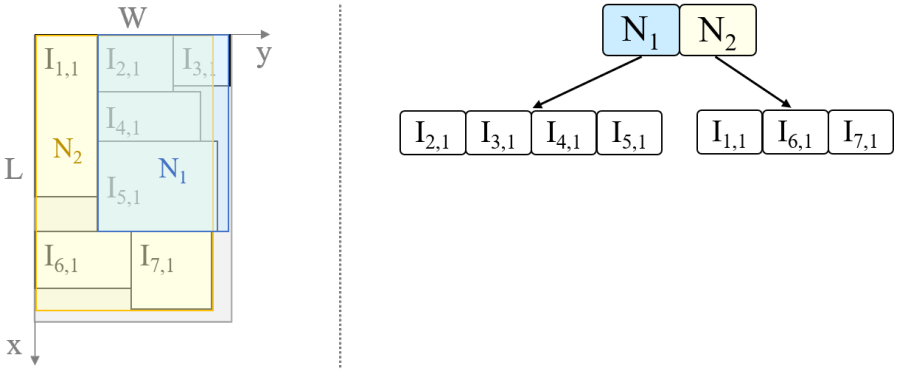
4.2.3 Deepest-Bottom-Left-Fill with RTree

Regarding common approaches for collision-detection, e.g. in game development, tree structures are the preferred choice. Thus, in this approach, we use a so-called RTree for the non-overlapping check (Guttman 1984). Hereby, the tree contains geometric objects. The main idea is to group geometries that are close to each other inside of the tree. At each level of the tree, there is a fixed number of objects. Therefore, an RTree represents a balanced tree. At the leaf level, each geometry describes a single object. In the next higher level of the tree, a minimum bounding geometry is defined by its underlying objects. An RTree can be used to execute queries, e.g. to ensure non-overlapping of geometries: Since all objects lie within the minimal bounding geometry, an object that does not intersect with the bounding geometry can also not intersect with any of the contained underlying objects.
In this third approach and also as proposed by Allen et al. (2011), an RTree is used to store items and their positions and to check for non-overlapping of the current item with all already placed items. In Fig. I.5, we show two examples for packed items in a vehicle and the corresponding RTree representation. As described, the leaves of the RTree represent the already placed items while the

nodes are the minimum bounding rectangles of the underlying items. The number of objects per node is here set to 4. As Fig. I.5b shows, the minimum bounding rectangles are allowed to overlap.



(a) Example 1



(b) Example 2

Figure I.5: Packed items in vehicle and created RTree

In this approach, we extended the DBLF algorithm by Allen et al. (2011) by considering the rotation and fragility of items. Moreover, to speed up the algorithm, positions where an item cannot be placed are removed (see below). The algorithm is summarized in Alg. I.4.
Similar to DBLF with Points, the possible placement points are stored in the set SP, and three new points are created based on the currently placed item [11]. As in the previous two DBLF algorithms, for each item I_p [3], every available placement position [4] and allowed rotation are tested [5] until a feasible position for the current item I_p is found [7]. If the non-overlapping check by the RTree is positive and also the other loading constraints are fulfilled, the placement of item I_p is saved [8], and the item with its position is additionally stored in the RTree [9]. In contrast to the other algorithms, no sliding techniques are used. After saving the feasible position of the item, three new placement points are created based on the current item I_p (see DBLF with Points) and added to the set SP [11]. The set is sorted according to the DBL order [12]. Then, the minimum dimensions l_{min} and h_{min} of all unplaced items are searched as in DBLF with Spaces [13]. The purpose of the minimum dimensions is to remove points whose distance to the vehicle walls is too small to place any item [14-18].

4.2.4 Comparison

The differences between the DBLF algorithm variants are summarized in Table I.1. In DBLF with Points, a non-overlapping check between a possible item position and all placed items is necessary. In contrast to that, this is not necessary for DBLF with Spaces, since the dimensions of the space represent empty volumes and are adapted after each item placement. In DBLF with RTree, the items and their positions are stored in the tree, which is then used to ensure the non-overlapping

Algorithm I.4 Deepest-Bottom-Left-Fill with RTree

Input: Instance data
Output: Packing Plan PP_v, feasibility of R_v

```
initialize sorted sequence of unpacked items IS
initialize set of unique available points SP
for each item I_p ∈ IS do
    for each point po ∈ SP do
        for each permitted rotation do
            if I_p does not overlap (RTree Query) then
                if position is feasible w.r.t. all loading constraints then
                    save placement for I_p
                    add item I_p in RTree
                    erase point sp
                    create and include new points in set SP
                    sort SP based on DBL
                    get l_min and h_min of unplaced items ∈ IS
                    for each point pi ∈ SP do
                        if pi too small then
                            erase space pi
                        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
```

condition. Moreover, in each algorithm, the smallest dimensions are searched to remove impossible placement positions. In the case of DBLF with Spaces and with RTree, the dimensions of the possible placement must be larger and higher than the smallest dimensions of all unplaced items in the route. The dimensions are determined after each placement of an item. In the DBLF with Points, the smallest dimensions are searched once in all items of the instance.

Table I.1: Comparison of DBLF approaches

	DBLF with		
	Points	Spaces	RTree
Non-overlapping check	Each item with all placed items	Spaces ensure non-overlapping	Using Tree
Sliding of items	After selecting placement point	While space creation	None
Smallest Dimensions	all items of Instance	unplaced items	unplaced items

5 Computational Experiments

This section presents the computational studies. In particular, we compare the three DBLF algorithm variants concerning solution quality and performance. Moreover, the impact of the loading constraints per DBLF algorithm variant is analyzed. Hereby, the instance sets by Zhang et al. (2017) as well as our 3L-VRPTW instance set[1] are used. Each instance and each constraint

[1] see http://open-science.ub.ovgu.de/xmlui/bitstream/handle/684882692/59/Overview.zip

set are tested 15 times. Out of these 15 runs per instance and constraint set, the best one is determined (lowest total travel distance, then lowest runtime), and the average result is presented as well. In B, we provide summarized result tables. Moreover, to ensure full transparency and traceability, we published all results on GitHub[2]. These are additionally validated with our solution validator[3].
The algorithms are implemented in C++ as a single-core application and are compiled using the VC++ 2019 version, v14.26 compiler. The experiments were executed on several i7-2600 quad-cores with 3.4 GHz and 16 GB RAM. The operating system is Windows 10. The RTree is implemented by using the well-known Boost library, 1.73 version[4].

5.1 Parameters

The necessary parameters for the hybrid algorithm are listed in Table I.2. 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 are kept. The *Number of Leaves per Node* was received experimentally.

Table I.2: Algorithm Parameters

Category	Description	Value
Stopping Criterion	Maximal number of iterations	25,000
Stopping Criterion	Maximal number of iterations without improvement	8,000
Stopping Criterion	Time limit [min]	60
ALNS	Number of iterations for updating probabilities for removal and insertion operators it_p	100
ALNS, Simulated Annealing	Cooling Rate γ	0.99975
ALNS, Simulated Annealing	Start Temperature Parameter w	5%
RTree	Number of Leaves per Node	16
Minimal Supporting Ratio	α	0.75

5.2 Comparison of DBLF Algorithms

In the following, we evaluate the solution quality and the performance of the DBLF algorithm variants. Two computational experiments are conducted. The first evaluates the algorithm variants based on predefined routes with given volume ranges. In the second, the algorithms are tested in the context of the hybrid algorithm.

5.2.1 Predefined Routes

For the following computational experiments, predefined routes for each instance of our instance set are created. The corresponding volume of all dispatched items in the route is calculated and set in relation to the vehicle loading space. Each route is tested once by each algorithm including all loading constraints. In Table I.3, the results are presented, including the success rate, which is the total number of successfully packed routes related to the total number of routes. Moreover, the average runtime per volume range is presented.

As expected, the higher the volume utilization rate within the vehicle loading space, the lower the success rate of the packing process. Surprisingly, the industrial approach (DBLF with RTree) achieves the worse results. The overall highest success rate is achieved by DBLF with Spaces, closely followed by DBLF with Points. However, the average runtime of the DBLF with Points is around 2.4 times the average runtime of the DBLF with Spaces. This shows the lower complexity (and thus the better concept of the non-overlapping check) of the DBLF with Spaces compared to the traditional approach (DBLF with Points). The success rate is an indicator of the efficiency of

[2]see https://github.com/CorinnaKrebs/Results
[3]see https://github.com/CorinnaKrebs/SolutionValidator
[4]see https://www.boost.org/users/history/version_1_73_0.html

Table I.3: Comparison of DBLF Algorithms with predefined routes

	Number of Routes per Range	Success Rate [%] DBLF with Points	Spaces	RTree	avg. time [s] DBLF with Points	Spaces	RTree
30-40	43.970	**98.51**	98.43	96.37	2.28	**1.18**	5.03
40-50	57.826	94.93	**95.61**	86.57	5.00	**2.87**	13.56
50-60	48.648	88.04	**88.21**	69.14	20.10	**12.48**	37.56
60-70	29.908	**83.90**	83.10	60.74	62.66	**34.41**	82.36
70-80	14.103	79.64	**81.17**	58.25	142.16	**66.75**	225.51
80-90	3.534	60.98	**62.85**	37.95	699.01	**272.61**	608.16
Total	197.989	84.33	**84.90**	68.17	155.20	**65.05**	162.03

sliding items: In DBLF with RTree, the items are not slid, and this approach achieves significantly worse results than the other approaches containing sliding techniques. Concerning the performance of the DBLF with RTree, the average runtime is almost 2.5 times the average runtime of DBLF with Spaces. Further analysis shows that the tree must be adapted after each successful placement of an item causing high runtimes. To conclude, the DBLF with Spaces achieves merely the highest success rate per volume range along with the smallest runtime.

5.2.2 Combination with Vehicle Routing Problem

Table I.4 gives an overview of the results per instance set for each DBLF algorithm. All loading constraints described in Sec. 3 are considered. Per instance set, we show the average total travel distance (*ttd*), the average runtime in seconds (*time*), and the average number of iterations (*iterations*) calculated based on all instances. Since each instance is tested 15 times, the results are further grouped by best out of 15 runs and the calculated average of the 15 runs. Moreover, we count the number of instances in which a DBLF algorithm was the only one to find the best result along with the total number of best results found (# Best results found).

Table I.4: Results for DBLF Algorithms – all Constraints

		DBLF with Points	Spaces	RTree	Benchmark
Results for our Instances					
best	∅ *ttd*	1,167.34	**1,158.36**	1,263.25	
	∅ time [s]	2,174.97	**2,133.41**	2,963.55	
	∅ iterations	11,187.05	10,227.22	3,212.00	
avg.	∅ *ttd*	1,195.68	**1,172.33**	1,288.66	
	∅ time [s]	2,182.33	**2,136.27**	2,970.96	
	∅ iterations	11,065.92	10,306.46	3,204.22	
# Best results found		94 / 363	**230 / 497**	5 / 180	
Results for Zhang et al. (2017) Instances					
best	∅ *ttd*	783.20	**777.07**	854.18	965.74
	∅ time [s]	499.08	**476.46**	1,911.17	
	∅ iterations	17,399.59	17,176.70	7,161.26	
avg.	∅*ttd*	788.41	**781.51**	869.51	971.35
	∅ time [s]	501.28	**468.89**	1,946.59	1,163.16
	∅ iterations	17,590.99	17,099.25	7,857.70	
# Best results found		5 / 7	**19 / 20**	1 / 2	

Generally, the same findings as before can be drawn: DBLF with Spaces dominates the other DBLF algorithms, which is reflected by the smaller total travel distance, better performance (smaller runtime), and a higher number of best results found. This highlights the importance of the reduced complexity of the non-overlapping check. Compared to DBLF with Spaces, the total travel distance obtained with DBLF with Points is longer. In the case of our instance set, the total travel distance increases by about 2%, while for Zhang et al. (2017) instances, it is $< 1\%$ on average. Therefore, it can be concluded that sliding the possible placement positions during their creation (DBLF with

Spaces) leads to better results than choosing one position and sliding the items afterward (DBLF with Points). In general, the solutions are achieved with an approximately 5% increase in runtime when using DBLF with Points.
Surprisingly, the results for DBLF with RTree are generally worse. On average, the total travel distance increases by up to 10% compared to DBLF with Spaces. In general, the runtime is also significantly higher, e.g. for our instances, it is approx. 39%, and for Zhang et al. (2017), it is even up to 3.2 times higher than with DBLF with Spaces. As explained before, the higher runtime is caused by the necessary adaptions of the structure of the RTree after each successful placement of an item. The check for overlap itself is much faster compared to the other algorithms. Due to these adaptions of the RTree, the time per iteration is higher and thus, fewer iterations can be conducted within the time limit. Another reason for the longer total travel distance lies in the missing sliding techniques used to place items in the deepest, leftmost, bottommost position. Still, the DBLF with RTree finds the best results for 182 of 627 instances (29%), and for 6 instances, DBLF with RTree is the only algorithm finding the best solution at all. Consequently, this approach has potential once the high runtime for adapting the tree structure is reduced. This could be achieved by other tree structures in future work.
In comparison to the benchmark by Zhang et al. 2017, DBLF with Spaces achieves solutions that save approximately 20% of total travel distance, on average, in 40% of the time. When comparing the best results with the average results for DBLF with Spaces, the total travel distance increases by up to 1.5%, on average, depending on the instance. Moreover, also the runtime varies by several additional percent. Consequently, there is still room for improvement to make the algorithm more stable in finding the best solutions. Nevertheless, DBLF with Spaces achieves the best results and has the best performance.

5.3 Impact of Loading Constraints

Above, each instance set was tested 15 times for each DBLF algorithm variant while all loading constraints are fulfilled. In this section, we want to evaluate the sensitivity of the DBLF algorithm variants concerning the loading constraints. Therefore, we analyze the impact of the loading constraints for each variant by removing one loading constraint and testing each instance of our instance set again 15 times. As before, we report results concerning the total travel distance (*ttd*), the runtime (*time*), and the number of iterations. The relative deviation of the results where one loading constraint is excluded to the results considering all constraints with the same DBLF algorithm variant is calculated. These deviations are further visualized by boxplots showing the minimum, 1st quartile (Q_1), median, 3rd quartile (Q_3), and maximum values of the relative results. Each boxplot is overlapped by a swarm plot indicating the distribution of the data points.
In general, the number of performed iterations varies greatly for all results. On the one hand, since the performance increases, more iterations can be carried out when disregarding one constraint. Then, also the runtime increases while the total travel distance does not necessarily improve. On the other hand, the number of performed iterations along with the runtime can decrease because the solution space is larger and thus, a good solution can be found faster and the algorithm terminates earlier.

Fig. I.6 visualizes the results for the instances when ignoring the rotation constraint (L3). Consequently, the solution space is smaller without allowing the rotation of items. However, for most instances, the total travel distance remains mostly the same independent of the DBLF algorithm, as Q_1 to the median are around zero. For one-quarter of the instances (Q_3), the total travel distance increases by several percent – especially for DBLF with Spaces by up to 2.6%. In the case of DBLF with Spaces, the median runtime reduction is around 16%, in the case of DBLF with Points even 32%. In the case of DBLF with RTree, the runtime remains unchanged. Unlike as stated in Kang et al. (2012), who tackled the pure 3D container loading problem, we can conclude that the rotation constraint has merely only small impacts on the objective value of the 3L-VRPTW.

In Fig. I.7, the results obtained without taking the load capacity (L4) into account are shown. Since the range between median to Q_3 is around zero, the constraint has merely no effect on the total travel distance. For DBLF with Points, for one quarter (Q_3), the total travel distance decreases by up to 7.4%. In the case of DBLF with Spaces and DBLF with Points, the runtime decreases

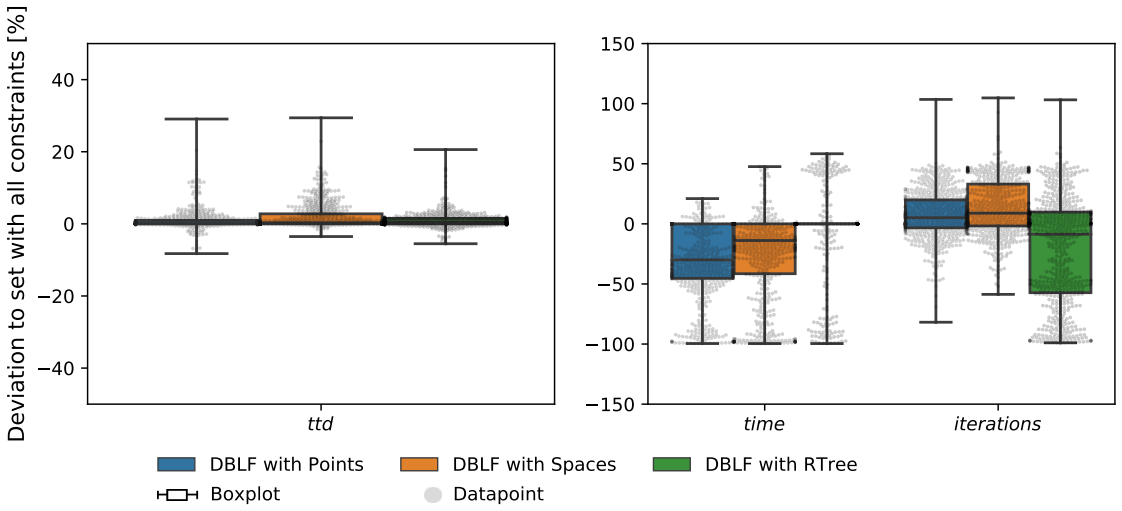


Figure I.6: Comparison of DBLF Results – Without Rotation

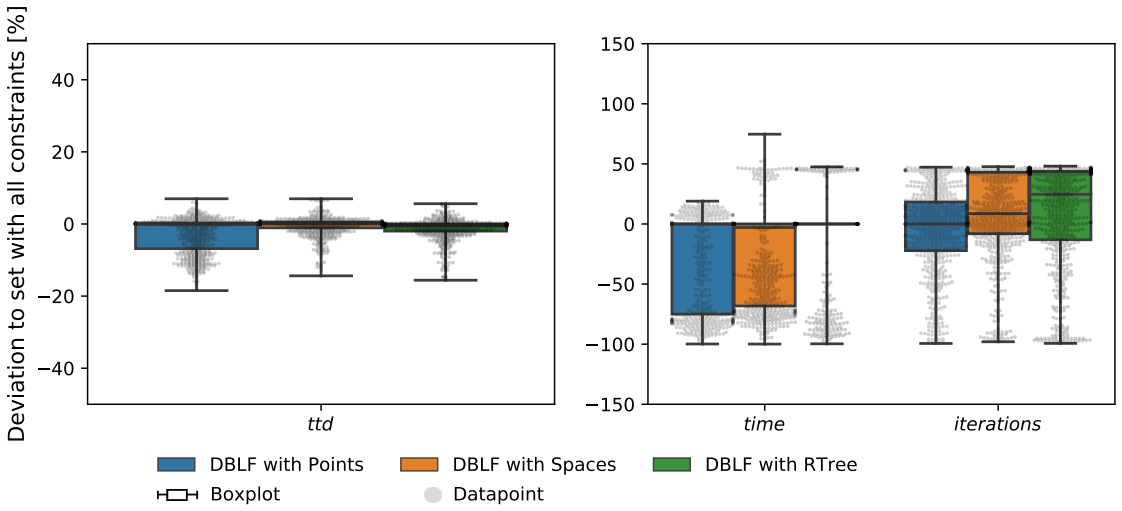


Figure I.7: Comparison of DBLF Results – Without Load Capacity

significantly (Q_1 up to -75%, $Q_3 = 0$); for DBLF with RTree, the runtime remains similar. The number of iterations varies widely for all algorithms (Q_1 = -19%, Q_3 = 43%). For our instances with a high number of customers, items, and item types, the solutions are generated faster with significantly fewer iterations. To conclude, the load capacity constraint has merely small impacts on the objective values and highly positive effects on the runtime.

Fig. I.8 illustrates the results without considering the LIFO policy (L5). The total travel distance improves significantly for almost every instance ($Q_3 = 0$, Q_1 up to -8.3%). The runtime decreases significantly for DBLF with Points ($Q_1 \approx -62\%$) and especially for DBLF with Spaces ($Q_1 \approx -68\%$). For the DBLF with RTree, there are only isolated improvements in runtime (Q_1 to $Q_3 = 0$). To summarize, the LIFO constraint has a significant impact on the total travel distance and the performance.

Fig. I.9 shows the impact of disregarding the minimal supporting area constraint (L6). The total travel distance remains the same (median and Q_3 are around zero) or decreases ($Q_1 \approx -4.8\%$). The runtime shows the same tendencies: Q_1 is -56% for DBLF with Points and -54% for DBLF with Spaces. Further analysis shows that improvements of the total travel distance are especially achieved for instances with a high number of different item types since the constraint is more restrictive for these instances than for instances with homogenous items.

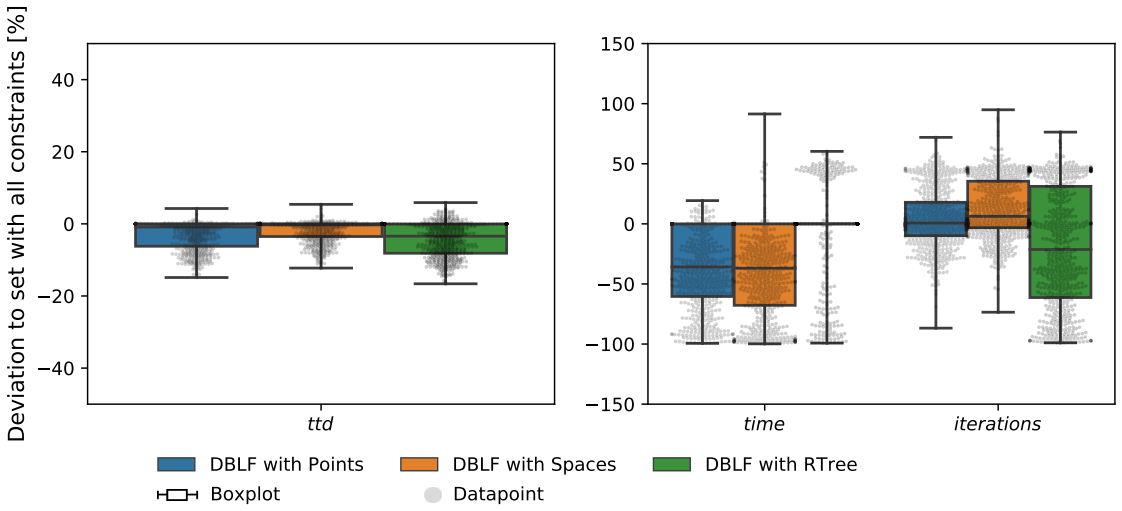


Figure I.8: Comparison of DBLF Results – Without LIFO

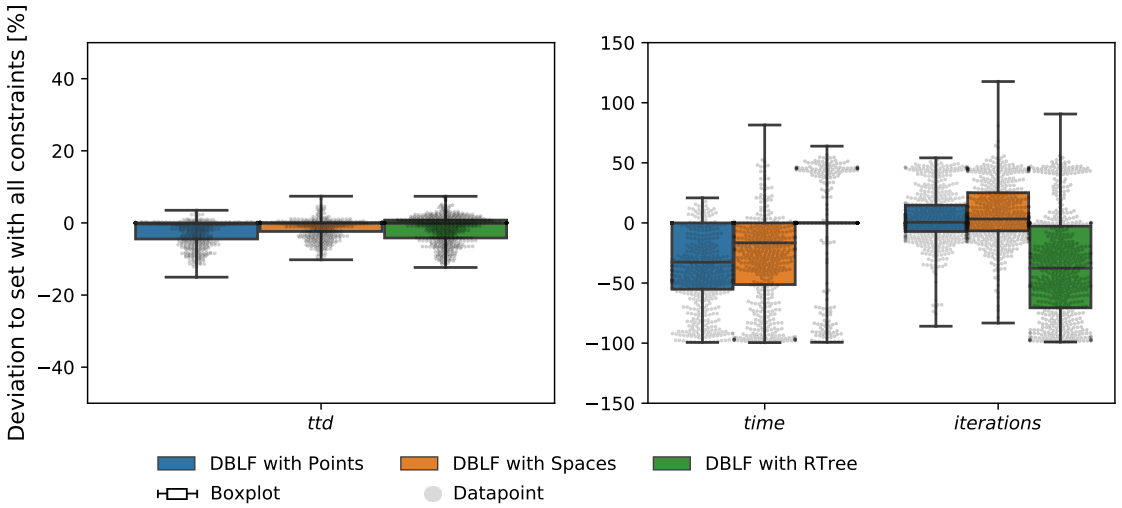


Figure I.9: Comparison of DBLF Results – Without Minimal Supporting Area

Fig. I.10 visualizes the results without the consideration of an item's fragility (L7). The total travel distance remains unchanged for most instances or improves slightly ($Q_1 \approx -2\%$, $Q_3 \approx 0.4\%$). The runtime decreases significantly ($Q_3 = 0$, Q_1 up to 56%) for DBLF with Points and DBLF with Spaces. In the case of DBLF with RTree, there is no effect on the runtime.

In summary, disregarding the rotation of items and the fragility of items have rather small impacts on the total travel distance. However, load capacity, minimal supporting area, and the LIFO constraint show significant effects. When disregarding a constraint, the runtime for DBLF with Points and with Spaces decreases significantly. To conclude, when designing loading algorithms, the focus should be on the fulfillment of the LIFO policy and minimal supporting area to improve the performance. For our future work, we plan to improve the DBLF with Spaces in this respect.

6 Conclusion

The DBLF algorithm introduced by Karabulut and İnceoğlu (2005) is a widely-used loading algorithm and the basis of this paper. In the context of the 3L-VRPTW, the DBLF algorithm

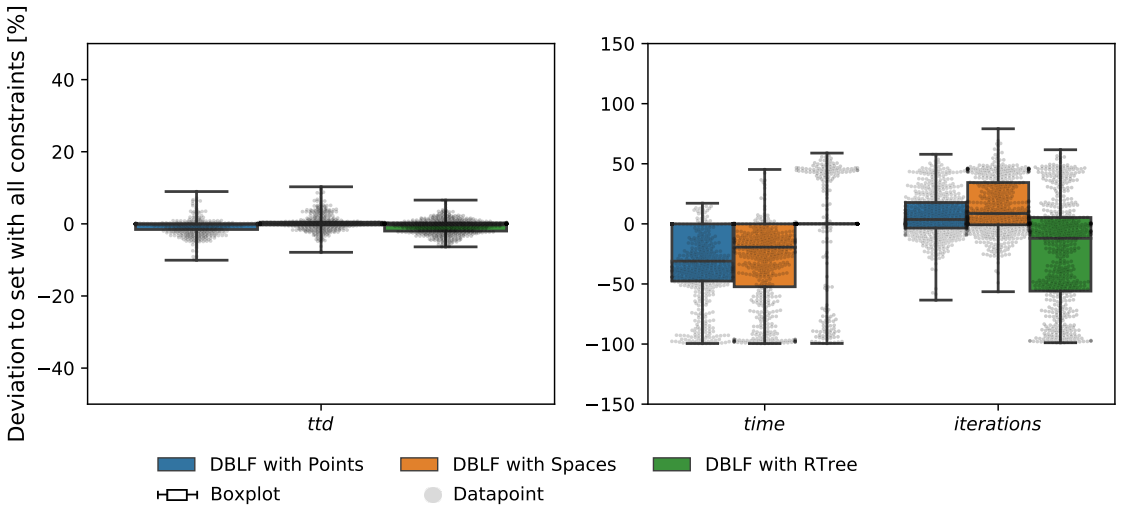


Figure I.10: Comparison of DBLF Results – Without Fragility

tackles the packing problem and finds feasible packing plans for a set of routes. Comparing the effectiveness of the investigated DBLF algorithm variants in this paper, our experiments show that DBLF with Spaces clearly finds superior results (shortest total travel distance) with the best performance (smallest runtime). Moreover, it outperforms the current benchmark by Zhang et al. (2017). Surprisingly, DBLF with RTree has a relatively high runtime caused by the necessary frequent restructuring of the tree. Therefore, other tree structures in combination with sliding techniques are worth to be examined further.
We also analyzed the impact of loading constraints per DBLF algorithm variant. The load capacity, the LIFO policy, and the minimal supporting area influence the total travel distance strongly. Therefore, based on the DBLF with Spaces variant, our future work will focus on the design and implementation of loading algorithms which are specialized in the fast fulfillment of these loading constraints. Despite the increased solution space, the results show that disregarding the rotation of items improves the total travel distance only slightly for most instances. Although the non-consideration of the fragility of items only has a minor impact on the total travel distance, for security reasons, this should be investigated further. As future work, we want to extend the problem w.r.t. a heterogeneous fleet.

Acknowledgements

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

References

Allen, S., Burke, E., and Kendall, G. (2011). “A hybrid placement strategy for the three-dimensional strip packing problem”. In: *European Journal of Operational Research* vol. 209, no. 3, pp. 219–227. ISSN: 0377-2217. DOI: 10.1016/j.ejor.2010.09.023. URL: https://doi.org/10.1016/j.ejor.2010.09.023.

Araújo, L. J., Panesar, A., Özcan, E., Atkin, J., Baumers, M., and Ashcroft, I. (2019). “An experimental analysis of deepest bottom-left-fill packing methods for additive manufacturing”. In: *International Journal of Production Research* vol. 0, no. 0, pp. 1–17. DOI: 10.1080/00207543.2019.1686187. URL: https://doi.org/10.1080/00207543.2019.1686187.

Baker, B., Coffman, E., and Rivest, R. (1980). “Orthogonal Packings in Two Dimensions”. In: *SIAM Journal on Computing* vol. 9, no. 4, pp. 846–855. DOI: 10.1137/0209064.

Clarke, G. and Wright, J. W. (1964). “Scheduling of Vehicles from a Central Depot to a Number of Delivery Points”. In: *Operations Research* vol. 12, no. 4, pp. 568–581. ISSN: 0030364X, 15265463. DOI: 10.1287/opre.12.4.568. URL: http://www.jstor.org/stable/167703.

Feng, X., Moon, I., and Shin, J. (2015). "Hybrid genetic algorithms for the three-dimensional multiple container packing problem". In: *Flexible Services and Manufacturing Journal* vol. 27, pp. 451–477. DOI: 10.1007/s10696-013-9181-8.

Gendreau, M., Iori, M., Laporte, G., and Martello, S. (2006). "A Tabu Search Algorithm for a Routing and Container Loading Problem". In: *Transportation Science* vol. 40, no. 3, pp. 342–350. ISSN: 0041-1655. DOI: 10.1287/trsc.1050.0145. URL: http://pubsmisc.informs.org/doi/abs/10.1287/trsc.1050.0145.

Guttman, A. (June 1984). "R-Trees: A Dynamic Index Structure for Spatial Searching". In: *SIGMOD Rec.* vol. 14, no. 2, pp. 47–57. ISSN: 0163-5808. DOI: 10.1145/971697.602266.

Hopper, E. and Turton, B. (2001). "An empirical investigation of meta-heuristic and heuristic algorithms for a 2D packing problem". In: *European Journal of Operational Research* vol. 128, no. 1, pp. 34–57. ISSN: 0377-2217. DOI: 10.1016/S0377-2217(99)00357-4. URL: https://doi.org/10.1016/S0377-2217(99)00357-4.

Jamrus, T. and Chien, C.-F. (2016). "Extended priority-based hybrid genetic algorithm for the less-than-container loading problem". In: *Computers & Industrial Engineering* vol. 96, pp. 227–236. ISSN: 0360-8352. DOI: https://doi.org/10.1016/j.cie.2016.03.030. URL: http://www.sciencedirect.com/science/article/pii/S036083521630105X.

Kang, K., Moon, I., and Wang, H. (2012). "A hybrid genetic algorithm with a new packing strategy for the three-dimensional bin packing problem". In: *Applied Mathematics and Computation* vol. 219, no. 3, pp. 1287–1299. ISSN: 0096-3003. DOI: 10.1016/j.amc.2012.07.036. URL: http://www.sciencedirect.com/science/article/pii/S0096300312007369.

Karabulut, K. and İnceoğlu, M. M. (2005). "A Hybrid Genetic Algorithm for Packing in 3D with Deepest Bottom Left with Fill Method". In: *Advances in Information Systems*. Ed. by Yakhno, T. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 441–450. ISBN: 978-3-540-30198-1.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). "Optimization by simulated annealing." In: *Science* vol. 220 4598, pp. 671–80. DOI: 10.1126/science.220.4598.671.

Koch, H. (2018). "Vehicle routing problems with three-dimensional loading constraints and backhauls". PhD thesis. Otto-von-Guericke-Universität Magdeburg. URL: http://dx.doi.org/10.25673/13615.

Koch, H., Bortfeldt, A., and Wäscher, G. (Feb. 2018). "A hybrid algorithm for the vehicle routing problem with backhauls, time windows and three-dimensional loading constraints". In: *OR Spectrum* vol. 40. DOI: 10.1007/s00291-018-0506-6.

Koch, H., Schlögell, M., and Bortfeldt, A. (Oct. 2019). "A hybrid algorithm for the vehicle routing problem with three-dimensional loading constraints and mixed backhauls". In: *Journal of Scheduling*. DOI: 10.1007/s10951-019-00625-7.

Krebs, C. and Ehmke, J. F. (2021a). "Axle Weights in combined Vehicle Routing and Container Loading Problems". In: *EURO Journal on Transportation and Logistics* vol. 10, p. 100043. ISSN: 2192-4376. DOI: 10.1016/j.ejtl.2021.100043. URL: https://www.sciencedirect.com/science/article/pii/S2192437621000157.

— (2021b). "Vertical Stability Constraints in Combined Vehicle Routing and 3D Container Loading Problems". In: *Computational Logistics*. Ed. by Mes, M., Lalla-Ruiz, E., and Voß, S. Cham: Springer International Publishing, pp. 442–455. ISBN: 978-3-030-87672-2.

Krebs, C., Ehmke, J. F., and Koch, H. (Aug. 2021). "Advanced loading constraints for 3D vehicle routing problems". In: *OR Spectrum*. ISSN: 1436-6304. DOI: 10.1007/s00291-021-00645-w. URL: https://doi.org/10.1007/s00291-021-00645-w.

Ma, H.-w., Zhu, W., and Xu, S. (2011). "Research on the Algorithm for 3L-CVRP with Considering the Utilization Rate of Vehicles". In: *Intelligent Computing and Information Science*. Ed. by Chen, R. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 621–629. ISBN: 978-3-642-18129-0.

Mak-Hau, V., Moser, I., and Aleti, A. (2018). "An Exact Algorithm for the Heterogeneous Fleet Vehicle Routing Problem with Time Windows and Three-Dimensional Loading Constraints". In: *Data and Decision Sciences in Action*. Ed. by Sarker, R., Abbass, H. A., Dunstall, S., Kilby, P., Davis, R., and Young, L. Cham: Springer International Publishing, pp. 91–101. ISBN: 978-3-319-55914-8. DOI: 10.1007/978-3-319-55914-8.

Moon, I. and Nguyen, L. (Oct. 2013). "Container packing problem with balance constraints". In: *OR Spectrum* vol. 36. DOI: 10.1007/s00291-013-0356-1.

Pace, S., Turky, A., Moser, I., and Aleti, A. (2015). "Distributing Fibre Boards: A Practical Application of the Heterogeneous Fleet Vehicle Routing Problem with Time Windows and Three-dimensional Loading Constraints". In: *Procedia Computer Science* vol. 51. International Conference On Computational Science, ICCS 2015, pp. 2257–2266. ISSN: 1877-0509. DOI: 10.1016/j.procs.2015.05.382. URL: http://www.sciencedirect.com/science/article/pii/S1877050915011904.

Ropke, S. and Pisinger, D. (2006). "A unified heuristic for a large class of Vehicle Routing Problems with Backhauls". In: *European Journal of Operational Research* vol. 171, no. 3. Feature Cluster: Heuristic and Stochastic Methods in Optimization Feature Cluster: New Opportunities for Operations Research, pp. 750–775. ISSN: 0377-2217. DOI: 10.1016/j.ejor.2004.09.004. URL: http://www.sciencedirect.com/science/article/pii/S0377221704005831.

Silva, E. F., Toffolo, T. A. M., and Wauters, T. (2019). "Exact methods for three-dimensional cutting and packing: A comparative study concerning single container problems". In: *Computers & Operations Research* vol. 109, pp. 12–27. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2019.04.020. URL: https://www.sciencedirect.com/science/article/pii/S0305054819301030.

Solomon, M. M. (1987). "Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints". In: *Operations Research* vol. 35, no. 2, pp. 254–265. DOI: 10.1287/opre.35.2.254.

Wauters, T., Verstichel, J., and Vanden Berghe, G. (2013). "An effective shaking procedure for 2D and 3D strip packing problems". In: *Computers & Operations Research* vol. 40, no. 11, pp. 2662–2669. ISSN: 0305-0548. DOI: 10.1016/j.cor.2013.05.017. URL: https://doi.org/10.1016/j.cor.2013.05.017.

Wu, B., Lin, J.-g., and Dong, M. (2013). "Artificial Bee Colony Algorithm for Three-Dimensional Loading Capacitated Vehicle Routing Problem". In: *Proceedings of 20th International Conference on Industrial Engineering and Engineering Management*. Ed. by Qi, E., Shen, J., and Dou, R. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 815–825. ISBN: 978-3-642-40063-6.

Zhang, D., Cai, S., Ye, F., Si, Y.-W., and Nguyen, T. T. (2017). "A hybrid algorithm for a vehicle routing problem with realistic constraints". In: *Information Sciences* vol. 394-395, pp. 167–182. ISSN: 0020-0255. DOI: 10.1016/j.ins.2017.02.028.

Zhu, W., Qin, H., Lim, A., and Wang, L. (2012). "A two-stage tabu search algorithm with enhanced packing heuristics for the 3L-CVRP and M3L-CVRP". In: *Computers & Operations Research* vol. 39, no. 9, pp. 2178–2195. ISSN: 0305-0548. DOI: 10.1016/j.cor.2011.11.001. URL: https://doi.org/10.1016/j.cor.2011.11.001.

Appendix

A Mathematical Formulation

Based on Koch (2018), this section presents a mathematical formulation for the 3L-VRPTW. First, we introduce the decision variables followed by the formulations for the vehicle routing problem. The last subsection deals with the formulation of the loading constraints.

A.1 Decision Variables

There are four decision variables for the mathematical formulation. The first, shown in Eq. I.1, represents the routing decision, i.e. that a vehicle v connects customer i with customer j in period t. In period t, the vehicle v has left node i and has not yet reached the next node j. Consequently, in period $t = 0$, the vehicle v has left the depot and drives to the first customer.

$$\varepsilon_{i,j}^{t,v} = \begin{cases} 1, & \text{if vehicle } v \text{ drives directly from node } i \text{ to node } j \text{ in period } t, \\ 0, & \text{otherwise} \end{cases} \tag{I.1}$$

The next decision variable (Eq. I.2) represents the decision for a position (x, y, z) of an item $I_{i,k}$ demanded by customer i, placed inside of vehicle v in period t.

$$\pi_{x,y,z}^{i,k,t,v} = \begin{cases} 1, & \text{if the item } I_{i,k} \text{ of customer } i \text{ in period } t \text{ inside of vehicle } v \\ & \text{is placed with its minimal corner point at position (x, y, z),} \\ 0, & \text{otherwise} \end{cases} \tag{I.2}$$

The third decision variable, Eq. I.3, describes the decision for the rotation of item $I_{i,k}$ along the length-width plane (see rotation constraint (L3)).

$$\sigma_{i,k} = \begin{cases} 1, & \text{if the item } I_{i,k} \text{ is not rotated, e.g. the length } l_{i,k} \text{ is parallel} \\ & \text{to the x-axis,} \\ 0, & \text{otherwise} \end{cases} \tag{I.3}$$

The last decision variable (Eq. I.4) determines whether a point (x, y, z) is occupied by an item $I_{i,k}$. This is necessary for the formulation of several loading constraints.

$$\rho_{x,y,z}^{i,k,v} = \begin{cases} 1, & \text{if an item } I_{i,k} \text{ in vehicle } v \text{ occupies the point (x, y, z)} \\ 0, & \text{otherwise} \end{cases} \tag{I.4}$$

A.2 Vehicle Routing

This section deals with the mathematical formulation of the vehicle routing part. If applicable, we link the formulas to the constraints introduced before. Note that constraint S1, ensuring the feasibility of a solution, is guaranteed through the mathematical formulation as a whole.
The objective function is to minimize the total travel distance (ttd), see Eq. I.5.

$$\min ttd = \sum_{i \in N} \sum_{j \in N} \sum_{t \in N} \sum_{v=0}^{v_{used}} (d_{i,j} \cdot \varepsilon_{i,j}^{t,v}) \tag{I.5}$$

Eq. I.6 corresponds to the solution constraint S2, so that at least one vehicle and at most the number of available vehicles (v_{max}) are dispatched.

$$v_{used} \in [1, v_{max}] \tag{I.6}$$

The next equations deal with the creation of routes. Eq. I.7 and the following equations ensure that each customer is left exactly once, corresponding to S3. Eq. I.8 ensures the connectivity of each route. Eq. I.9 prevents the situation in which the vehicle travels to the same node again after leaving it. Each customer is visited and left by the same vehicle (Eq. I.10). The depot is left at most once in period 0 (Eq. I.11) and not later (Eq. I.12) in every route. These equations cover constraint C1, ensuring that each route starts and ends at the depot and visiting at least one customer.

$$\sum_{j \in N} \sum_{t \in N} \sum_{v=0}^{v_{used}} \varepsilon_{i,j}^{t,v} = 1 \qquad \forall i \in N \setminus \{0\} \tag{I.7}$$

$$\sum_{j \in N} \sum_{t \in N \setminus \{0\}} \sum_{v=0}^{v_{used}} (t \cdot \varepsilon_{i,j}^{t,v}) - \sum_{j \in N} \sum_{t \in N} \sum_{v=0}^{v_{used}} (t \cdot \varepsilon_{i,j}^{t,v}) = 1 \qquad \forall i, t \in N \setminus \{0\} \tag{I.8}$$

$$\varepsilon_{i,i}^{t,v} = 0 \qquad \forall i, t \in N, \forall v \tag{I.9}$$

$$\sum_{j \in N} \varepsilon_{i,j}^{t+1,v} - \sum_{j \in N} \varepsilon_{j,i}^{t,v} = 0 \qquad \forall i \in N \setminus \{0\}, \forall t \in N \setminus \{n\}, \forall v \tag{I.10}$$

$$\sum_{j \in N \setminus \{0\}} \varepsilon_{0,j}^{0,v} \leq 1 \qquad \forall v \tag{I.11}$$

$$\varepsilon_{0,j}^{t,v} = 0 \qquad \forall j \in N \setminus \{0\}, \forall t \in N \setminus \{n\}, \forall v \tag{I.12}$$

The following equations ensure the consideration of the time windows. Hereby, $start_i^v$ is the time when the vehicle v starts the unloading process at the customer i. M_1 is a sufficiently large number[5]. In Eq. I.13, the start time for each route is set to zero. Eq. I.14 ensures that the unloading

[5] e.g. $M_1 \geq \max_{i \in N} DD_i + \max_{i \in N} ST_i + \max_{i,j \in E} c_{i,j} - \min_{i \in N} RT_i$

of the items at customer i does not start before the arrival at the customer i. Eq. I.15 deals with the waiting times, resulting when the vehicle v arrives at a customer i before its ready time RT_i, which are considered also in the following calculations. Eq. I.16 guarantees that the start of the unloading process at customer i does not begin before the ready time RT_i. Due to Eq. I.15, the unloading process at customer i is not allowed to start after the due date DD_i. This corresponds to constraint C2.

$$d_{0,i} - start_i^v \leq M_1 \cdot (1 - \varepsilon_{0,i}^{0,v}) \qquad \forall i \in N \setminus \{0\}, \forall v \tag{I.13}$$

$$start_i^v + ST_i + d_{i,j} - start_j^v \leq M_1 \cdot (1 - \varepsilon_{i,j}^{t,v})$$
$$\forall i \in N \setminus \{0\}, \forall j \in N, \forall v, \forall t \in N \setminus \{0\} \tag{I.14}$$

$$start_j^v \geq start_i^v + ST_i + d_{i,j} \qquad \forall i \in N \setminus \{0\}, \forall j \in N, \forall v \tag{I.15}$$

$$start_i^v \geq RT_i \qquad \forall i \in N, \forall v \tag{I.16}$$

$$start_i^v \leq DD_i \qquad \forall i \in N, \forall v \tag{I.17}$$

A.3 Container Loading

This section deals with the mathematical formulation for the container loading part. We first introduce the required variables. After that, the formulation of the loading constraints follows. Please note that we assume integer values for positions and dimensions. Moreover, note that an item is positioned in its the rearmost, leftmost bottom corner point. These conditions guarantee the orthogonal packing of items, as expected in the Orthogonality constraint (L2).
The following equations deal with the possible positions of an item $I_{i,k}$ inside of a vehicle loading space. To prevent overlapping, an item's dimensions (length, width, height) reduces the possible positions. Eq. I.18 represents the available positions along the x-axis for an item $I_{i,k}$. As the rotation of an item $I_{i,k}$ along the length-width plane is allowed, the possible placement x-coordinates are reduced accordingly. The same applies to the possible y-coordinates, as stated in Eq. I.19. The possible z-coordinates are only reduced by the item's height $h_{i,k}$ (see Eq. I.20).

$$X_{i,k} = \{0, 1, 2, ..., L - \min(l_{i,k}, w_{i,k})\} \tag{I.18}$$

$$Y_{i,k} = \{0, 1, 2, ..., W - \min(l_{i,k}, w_{i,k})\} \tag{I.19}$$

$$Z_{i,k} = \{0, 1, 2, ..., H - h_{i,k}\} \tag{I.20}$$

Using the rotation decision variable (Eq. I.3), the length and width of an item $I_{i,k}$ can be further described as:

$$l'_{i,k} = \sigma_{i,k} \cdot l_{i,k} + (1 - \sigma_{i,k}) \cdot w_{i,k} \tag{I.21}$$

$$w'_{i,k} = (1 - \sigma_{i,k}) \cdot l_{i,k} + \sigma_{i,k} \cdot w_{i,k} \tag{I.22}$$

The placement position ($x_{i,k}$, $y_{i,k}$, $z_{i,k}$,) of an item $I_{i,k}$ can be determined by using the placement decision variable as shown in Eq. I.23-I.25:

$$x_{i,k} = \sum_{t \in N \setminus \{n\}} \sum_{v=0}^{v_{used}} \sum_{x \in X_{i,k}} \sum_{y \in Y_{i,k}} \sum_{z \in Z_{i,k}} x \cdot \pi_{x,y,z}^{i,k,t,v} \tag{I.23}$$

$$y_{i,k} = \sum_{t \in N \setminus \{n\}} \sum_{v=0}^{v_{used}} \sum_{x \in X_{i,k}} \sum_{y \in Y_{i,k}} \sum_{z \in Z_{i,k}} y \cdot \pi_{x,y,z}^{i,k,t,v} \tag{I.24}$$

$$z_{i,k} = \sum_{t \in N \setminus \{n\}} \sum_{v=0}^{v_{used}} \sum_{x \in X_{i,k}} \sum_{y \in Y_{i,k}} \sum_{z \in Z_{i,k}} z \cdot \pi_{x,y,z}^{i,k,t,v} \tag{I.25}$$

The occupancy of a point (x', y', z') by an item $I_{i,k}$ placed in point (x, y, z) is determined in the following way (see Eq. I.26):

$$\rho^{i,k,v}_{x',y',z'} = \sum_{t\in N\setminus\{n\}} \sum_{\{x\in X_{i,k}|x'-l'_{i,k}+1\leq x\leq x'\}} \sum_{\{y\in Y_{i,k}|y'-w'_{i,k}+1\leq y\leq y'\}} \sum_{\{z\in Z_{i,k}|z'-h_{i,k}+1\leq z\leq z'\}} \pi^{i,k,t,v}_{x,y,z} \tag{I.26}$$

Eq. I.27 guarantees that for every item $I_{i,k}$ of a customer i exactly one position is found inside of a vehicle v. Through Eq. I.28, it is guaranteed that all items of customer i are packed in the vehicle v. These two equations ensure the constraint S4.

$$\sum_{k=1}^{c_i} \sum_{t\in N\setminus\{n\}} \sum_{v=0}^{v_{used}} \sum_{x\in X_{i,k}} \sum_{y\in Y_{i,k}} \sum_{z\in Z_{i,k}} \pi^{i,k,t,v}_{x,y,z} = 1 \qquad \forall i \in N\setminus\{0\} \tag{I.27}$$

$$\sum_{k=1}^{c_i} \sum_{x\in X_{i,k}} \sum_{y\in Y_{i,k}} \sum_{z\in Z_{i,k}} \pi^{i,k,t,v}_{x,y,z} = c_i \cdot \sum_{i\in N} \varepsilon^{t,v}_{i,j} \qquad \forall i \in N\setminus\{0\}, \forall t \in N\setminus\{n\}, \forall v \tag{I.28}$$

Eq. I.29 ensures the non-overlapping of items by excluding the position points based on the set rotation.

$$\sum_{\{x\in X_{i,k}|x>L-l'_{i,k}\}} \sum_{y\in Y_{i,k}} \pi^{i,k,t,v}_{x,y,z} + \sum_{x\in X_{i,k}} \sum_{\{y\in Y_{i,k}|y>W-w'_{i,k}\}} \pi^{i,k,t,v}_{x,y,z} = 0$$
$$\forall i \in N\setminus\{0\}, \forall c_i, \forall v, \forall t \in N\setminus\{n\}, \forall z \in Z_{i,k} \tag{I.29}$$

Eq. I.30 presents an alternative formulation for Eq. I.29 where each point inside of the vehicle loading space is allowed to be occupied at most once. Combined with the previous definitions, this ensures the Geometry constraint (L1).

$$\sum_{i\in N\setminus\{0\}} \sum_{k}^{c_i} \rho^{i,k,v}_{x,y,z} \leq 1 \qquad \forall v, \forall x \in X_{i,k}, \forall y \in Y_{i,k}, \forall z \in Z_{i,k} \tag{I.30}$$

Eq. I.31 prevents exceeding of the vehicle loading space volume. To this end, the volumes of the loads for each customer i and each item $I_{i,k}$ are added up, and the sum must be smaller than the available loading space volume of the vehicle v. Eq. I.32 corresponds to the load capacity constraint L4, ensuring that the load capacity of vehicle v is not exceeded. Hereby, for each customer i and each item $I_{i,k}$, the load mass is added up. The sum must be smaller than the load capacity of the vehicle v.

$$\sum_{i\in N\setminus\{0\}} \sum_{k=1}^{c_i} \sum_{j\in N} \sum_{t\in N\setminus\{0\}} \varepsilon^{t,v}_{i,j} \cdot l_{i,k} \cdot w_{i,k} \cdot h_{i,k} \leq L \cdot W \cdot H \qquad \forall v \tag{I.31}$$

$$\sum_{i\in N\setminus\{0\}} \sum_{k=1}^{c_i} \sum_{j\in N} \sum_{t\in N\setminus\{0\}} \varepsilon^{t,v}_{i,j} \cdot m_{i,k} \leq D \qquad \forall v \tag{I.32}$$

Eqs. I.33 and I.34 ensure the LIFO constraint (L5). The first one guarantees the LIFO policy along the x-axis. It prevents positions where an item $I_{i,k}$, placed in point (x, y, z) and delivered in period t, would be behind another item $I_{j,q}$ placed in point (x', y', z') and delivered in period u, a period later than t ($u > t$). Consequently, Eq. I.33 sums the items $I_{j,q}$ delivered later than period t and being placed in front of item $I_{i,k}$ delivered in period t. The sum must be smaller than a sufficiently large number[6] M_3.

[6] e.g. $M_3 \geq \sum_{i\in N\setminus\{0\}} \cdot c_i$

$$\sum_{j \in N \setminus \{0\}} \sum_{q}^{c_j} \sum_{\{u \in N | t < u < n\}} \sum_{\{x' \in X_{j,q} | x' \geq x + l'_{i,k}\}} \sum_{\{y' \in Y_{j,q} | y - w'_{j,q} + 1 \leq y' \leq y + w'_{i,k} - 1\}} \sum_{\{z' \in Z_{j,q} | z - h_{j,q} + 1 \leq z' \leq z + h_{i,k} - 1\}} \cdot \pi^{j,q,u,v}_{x',y',z'} \leq (1 - \pi^{i,k,t,v}_{x,y,z}) \cdot M_3$$
$$\forall i \in N \setminus \{0\}, \forall c_i, \forall v, \forall t \in N \setminus \{n\}, \forall x \in X_{i,k}, \forall y \in Y_{i,k}, \forall z \in Z_{i,k} \tag{I.33}$$

The same procedure as shown in Eq. I.33 applies for Eq. I.34 concerning the z-axis. Consequently, it prevents placements where an item $I_{i,k}$ placed in point (x, y, z) and delivered in period t is underneath another item $I_{j,q}$ placed in point (x', y', z') and delivered in period u, a period later than t ($u > t$).

$$\sum_{j \in N \setminus \{0\}} \sum_{q}^{c_j} \sum_{\{u \in N | t < u < n\}} \sum_{\{x' \in X_{j,q} | x - l'_{j,q} + 1 \leq x' \leq x + l'_{i,k} - 1\}} \sum_{\{y' \in Y_{j,q} | y - w'_{j,q} + 1 \leq y' \leq y + w'_{i,k} - 1\}} \sum_{\{z' \in Z_{j,q} | z' \geq z + h_{i,k}\}} \cdot \pi^{j,q,u,v}_{x',y',z'} \leq (1 - \pi^{i,k,t,v}_{x,y,z}) \cdot M_3$$
$$\forall i \in N \setminus \{0\}, \forall c_i, \forall v, \forall t \in N \setminus \{n\}, \forall x \in X_{i,k}, \forall y \in Y_{i,k}, \forall z \in Z_{i,k} \tag{I.34}$$

The minimal supporting area constraint (L6) is formulated in Eq. I.35. Hereby, the sum of all points which lay directly underneath of item $I_{i,k}$ and are occupied by another item is determined. This sum must be equal or greater than the base area of item $I_{i,k}$ multiplied by the support parameter α.

$$\sum_{j \in N \setminus \{0\}} \sum_{q}^{c_j} \sum_{\{x' \in X_{j,q} | x \leq x' \leq x + l'_{i,k} - 1\}} \sum_{\{y' \in Y_{j,q} | y \leq y' \leq y + w'_{i,k} - 1\}} \rho^{j,q,v}_{x',y',z-1} \geq \alpha \cdot l_{i,k} \cdot w_{i,k} \cdot \pi^{i,k,t,v}_{x,y,z}$$
$$\forall i \in N \setminus \{0\}, \forall c_i, \forall v, \forall t \in N \setminus \{n\}, \forall x \in X_{i,k}, \forall y \in Y_{i,k}, \forall z \in Z_{i,k} \setminus \{0\} \tag{I.35}$$

Eq. I.36 corresponds to the fragility constraint (L7), where non-fragile items cannot be placed on top of fragile items. However, fragile items are allowed to stack on top of every item. Note that the fragility flag for fragile items is 1. Suppose that item $I_{i,k}$ is placed on top of another item $I_{j,q}$. Then, the top surface of item $I_{j,q}$ touches the bottom of item $I_{i,k}$. The corresponding fragility flags of the underlying items are added up. This is expressed by the left side of the equal sign. The other side expresses the number of allowed fragile items placed underneath item $I_{i,k}$. If item $I_{i,k}$ is fragile, it is M_2 (sufficient large number[7]). If item $I_{i,k}$ is non-fragile, no fragile items are allowed to be placed underneath of item $I_{i,k}$ (= zero).

$$\sum_{j \in N \setminus \{0\}} \sum_{q}^{c_j} \sum_{\{u \in N | t \leq u < n\}} \sum_{\{x' \in X_{j,q} | x - l'_{j,q} + 1 \leq x' \leq x + l'_{i,k} - 1\}} \sum_{\{y' \in Y_{j,q} | y - w'_{j,q} + 1 \leq y' \leq y + w'_{i,k} - 1\}} f_{j,q} \cdot \pi^{j,q,u,v}_{x',y',z-h_{j,q}} \leq (1 - (1 - f_{i,k}) \cdot \pi^{i,k,t,v}_{x,y,z}) \cdot M_2$$
$$\forall i \in N \setminus \{0\}, \forall c_i, \forall v, \forall t \in N \setminus \{n\}, \forall x \in X_{i,k}, \forall y \in Y_{i,k}, \forall z \in Z_{i,k} \setminus \{0\} \tag{I.36}$$

B Result Tables

[7] e.g. $M_2 \geq \sum_{i \in N \setminus \{0\}} \cdot c_i$

Table I.5: Results for DBLF with Points – All Constraints

Instances			*∅ttd*		∅ time [s]		∅ iterations	
n	m	item types	best	avg.	best	avg.	best	avg.
20	200	3	408.62	409.13	589.39	592.61	7,844.40	7,937.79
		10	417.03	417.88	1,804.29	1,804.56	5,215.70	5,264.47
		100	422.22	422.61	1,805.44	1,828.12	7,482.55	7,482.71
	400	3	440.27	443.71	2,071.40	2,096.03	5,626.25	5,703.95
		10	459.59	466.71	2,464.40	2,471.42	3,589.65	3,697.46
		100	482.60	489.77	2,678.65	2,674.07	3,746.65	4,061.67
60	200	3	1,004.94	1,014.13	1,525.79	1,540.68	15,011.75	14,440.42
		10	1,030.97	1,055.00	1,654.37	1,665.33	13,653.45	13,415.70
		100	1,069.41	1,097.51	1,615.52	1,628.51	14,002.48	13,892.59
	400	3	1,341.82	1,367.85	1,957.30	1,962.52	11,383.23	10,963.66
		10	1,355.07	1,404.86	2,372.11	2,381.11	10,748.58	11,044.82
		100	1,473.33	1,496.02	2,893.46	2,903.03	10,405.80	10,332.49
100	200	3	1,194.90	1,207.99	2,164.28	2,162.31	15,070.53	14,427.46
		10	1,271.18	1,297.21	2,265.07	2,270.37	14,603.18	14,534.88
		100	1,358.49	1,388.50	2,109.06	2,109.28	15,031.83	14,604.46
	400	3	1,628.59	1,668.93	2,672.88	2,683.39	10,368.45	9,981.93
		10	1,726.92	1,793.91	2,830.23	2,831.52	11,148.18	11,043.06
		100	1,739.25	1,818.47	2,857.75	2,863.59	9,625.73	10,233.30
average			1,167.34	1,195.68	2,174.97	2,182.33	11,187.05	11,065.92

Table I.6: Results for DBLF with Points – w/o Rotation

Instances			*∅ttd*		∅ time [s]		∅ iterations	
n	m	item types	best	avg.	best	avg.	best	avg.
20	200	3	408.91	409.16	467.79	465.98	8,068.85	8,104.62
		10	417.60	418.24	1,803.15	1,803.36	5,285.35	5,328.53
		100	422.22	423.37	948.70	977.51	7,951.25	8,002.39
	400	3	442.50	447.04	1,942.43	1,955.39	5,339.65	5,525.85
		10	463.51	470.18	2,409.62	2,398.18	3,586.60	3,616.47
		100	486.00	492.11	2,262.74	2,271.69	3,967.65	4,084.61
60	200	3	1,010.53	1,027.45	1,012.94	1,020.00	14,247.93	14,318.14
		10	1,035.79	1,069.02	1,110.80	1,117.08	14,100.85	13,604.02
		100	1,078.05	1,112.55	913.13	919.39	14,172.00	13,603.87
	400	3	1,375.90	1,398.23	1,560.48	1,557.57	10,968.15	11,355.55
		10	1,396.63	1,461.18	1,232.17	1,236.06	10,673.05	10,638.23
		100	1,492.83	1,584.12	1,249.35	1,254.54	9,807.60	9,829.57
100	200	3	1,201.26	1,213.04	1,828.87	1,841.72	14,844.18	15,025.90
		10	1,267.13	1,286.97	1,993.51	2,005.06	16,028.05	15,163.26
		100	1,350.88	1,374.65	1,880.68	1,895.77	16,103.13	16,258.16
	400	3	1,654.60	1,695.88	2,100.85	2,104.17	10,762.50	10,885.11
		10	1,734.00	1,806.92	2,080.25	2,083.28	12,169.68	11,688.64
		100	1,744.42	1,832.04	2,115.35	2,121.38	10,841.28	10,981.78
average			1,177.49	1,212.81	1,599.71	1,606.14	11,454.54	11,378.90

Table I.7: Results for DBLF with Points – w/o LIFO

Instances			*∅ttd*		∅ time [s]		∅ iterations	
n	m	item types	best	avg.	best	avg.	best	avg.
20	200	3	408.21	408.49	369.68	377.67	7,970.65	8,078.55
		10	413.89	414.39	1,446.72	1,478.37	5,393.75	5,274.66
		100	419.71	420.49	786.30	795.92	6,811.95	6,939.82
	400	3	436.60	437.60	1,076.99	1,089.22	5,740.00	5,735.48
		10	445.59	447.90	2,258.86	2,270.90	3,613.65	3,612.29
		100	461.28	465.67	2,256.39	2,260.58	3,951.80	3,849.71
60	200	3	989.50	997.84	917.76	926.67	13,610.85	14,036.38
		10	975.26	1,007.24	1,022.84	1,039.90	12,728.98	12,780.77
		100	1,002.12	1,036.89	786.19	812.06	14,241.28	13,886.50
	400	3	1,326.42	1,338.36	1,508.52	1,535.82	10,937.93	11,288.85
		10	1,289.83	1,335.88	1,446.92	1,454.15	10,145.23	10,558.46
		100	1,387.14	1,454.60	1,202.02	1,209.00	10,169.03	10,461.06
100	200	3	1,178.14	1,187.66	1,716.21	1,721.82	14,832.25	14,630.52
		10	1,209.19	1,226.96	1,908.55	1,905.60	15,465.08	14,883.87
		100	1,279.41	1,301.27	1,794.60	1,811.47	15,813.83	15,527.56
	400	3	1,597.43	1,631.16	2,102.26	2,112.30	10,625.73	10,646.79
		10	1,641.24	1,705.68	2,227.68	2,237.08	10,877.38	10,823.70
		100	1,617.09	1,693.67	2,233.12	2,238.92	9,911.18	9,525.06
average			1,119.03	1,147.63	1,530.94	1,542.74	11,073.31	11,052.98

Table I.8: Results for DBLF with Points – w/o Minimal Supporting Area

Instances			*∅ttd*		∅ time [s]		∅ iterations	
n	m	item types	best	avg.	best	avg.	best	avg.
20	200	3	408.81	409.39	515.87	516.96	7,986.95	8,079.97
		10	416.85	417.59	1,692.97	1,700.14	5,355.60	5,405.89
		100	420.15	421.47	365.73	381.90	7,562.85	7,675.31
	400	3	439.78	443.21	1,831.21	1,851.53	5,425.75	4,959.65
		10	455.23	461.09	2,425.73	2,430.72	3,652.40	3,598.25
		100	469.13	473.62	2,257.86	2,263.88	4,029.20	3,884.22
60	200	3	998.22	1,010.48	1,039.74	1,054.92	13,874.70	13,725.37
		10	992.12	1,020.05	1,062.12	1,071.11	13,505.08	13,594.86
		100	1,013.58	1,041.18	836.14	826.08	15,485.28	14,444.97
	400	3	1,325.77	1,351.19	1,568.54	1,582.66	11,164.83	10,931.38
		10	1,307.44	1,354.69	1,394.26	1,401.98	10,140.78	10,652.61
		100	1,378.41	1,437.85	1,214.58	1,218.45	11,356.30	10,524.39
100	200	3	1,184.56	1,196.98	1,838.95	1,840.67	15,812.93	14,968.25
		10	1,226.99	1,248.10	1,954.39	1,971.51	14,817.60	14,577.91
		100	1,292.19	1,311.62	1,647.78	1,691.51	16,394.00	16,253.25
	400	3	1,612.72	1,649.00	2,094.65	2,100.07	10,799.80	10,445.15
		10	1,659.36	1,717.74	2,140.47	2,144.94	11,507.38	11,435.09
		100	1,631.82	1,699.09	2,181.10	2,189.20	9,621.73	10,236.14
average			1,128.54	1,156.74	1,567.83	1,577.71	11,432.45	11,239.40

Table I.9: Results for DBLF with Points – w/o Fragility

Instances			*∅ttd*		∅ time [s]		∅ iterations	
n	m	item types	best	avg.	best	avg.	best	avg.
20	200	3	408.78	409.15	475.00	489.61	8,067.60	8,297.08
		10	416.90	417.53	1,701.20	1,707.18	5,397.30	5,402.51
		100	420.52	422.51	475.05	502.02	8,003.25	7,819.21
	400	3	439.78	443.47	1,776.81	1,792.58	5,340.65	5,576.86
		10	457.73	464.42	2,419.38	2,422.82	3,725.75	3,619.69
		100	475.95	480.84	2,320.47	2,332.22	4,205.45	4,342.82
60	200	3	1,003.60	1,021.28	982.46	992.24	13,325.60	13,826.13
		10	1,014.12	1,046.33	1,103.21	1,116.92	13,990.50	13,863.84
		100	1,052.08	1,085.80	839.47	846.12	14,271.80	13,931.99
	400	3	1,332.43	1,356.79	1,554.15	1,567.07	11,188.58	10,968.21
		10	1,337.50	1,402.06	1,302.63	1,309.05	10,768.55	10,501.32
		100	1,440.80	1,524.92	1,148.59	1,155.42	10,370.58	10,101.56
100	200	3	1,186.87	1,198.90	1,858.05	1,869.46	15,294.35	14,697.74
		10	1,248.84	1,269.29	2,053.31	2,053.75	15,392.05	15,459.34
		100	1,321.72	1,345.33	1,818.43	1,824.73	15,991.73	16,102.71
	400	3	1,620.74	1,663.94	2,113.01	2,129.61	11,210.15	10,619.06
		10	1,696.96	1,764.17	2,154.56	2,154.60	11,632.85	11,561.21
		100	1,686.34	1,772.67	2,215.36	2,213.42	10,776.08	10,560.68
average			1,150.12	1,184.70	1,581.81	1,590.37	11,438.85	11,314.86

Table I.10: Results for DBLF with Points – w/o Load Capacity

Instances			*∅ttd*		∅ time [s]		∅ iterations	
n	m	item types	best	avg.	best	avg.	best	avg.
20	200	3	406.97	409.69	567.68	568.90	7,213.55	7,127.09
		10	416.99	417.86	1,452.03	1,452.44	5,416.75	5,405.27
		100	422.22	423.87	1,237.98	1,241.07	7,689.45	7,208.60
	400	3	439.77	441.31	1,826.25	1,839.18	3,963.70	3,861.44
		10	459.42	465.21	2,474.89	2,484.19	3,421.30	2,999.56
		100	481.63	487.48	2,697.56	2,686.80	3,029.95	2,645.97
60	200	3	995.66	999.06	1,172.50	1,174.24	13,933.75	13,537.70
		10	1,013.66	1,022.58	1,322.13	1,323.84	14,631.98	14,582.10
		100	1,049.25	1,056.89	1,488.99	1,482.64	15,312.65	15,034.73
	400	3	1,338.54	1,348.82	1,860.44	1,860.67	9,351.63	8,966.77
		10	1,355.94	1,370.59	2,039.40	2,043.99	11,258.18	11,056.39
		100	1,477.76	1,494.71	2,340.28	2,348.89	11,541.08	11,169.88
100	200	3	1,180.56	1,186.65	1,471.07	1,476.04	14,569.43	14,232.62
		10	1,250.06	1,258.45	1,745.19	1,748.69	15,638.30	15,661.47
		100	1,337.27	1,347.34	1,710.06	1,704.67	16,601.08	16,115.42
	400	3	1,626.05	1,647.85	2,546.67	2,552.41	9,204.53	9,110.51
		10	1,732.93	1,755.37	2,640.72	2,643.01	10,400.23	10,108.76
		100	1,735.57	1,765.63	2,637.99	2,631.00	9,477.10	9,964.76
average			1,160.45	1,171.78	1,873.58	1,875.09	11,152.48	10,944.34

Table I.11: Results for DBLF with Spaces – All Constraints

Instances			*∅ttd*		∅ time [s]		∅ iterations	
n	m	item types	best	avg.	best	avg.	best	avg.
20	200	3	408.37	408.71	570.65	572.66	7,831.90	7,624.40
		10	416.65	417.69	1,801.83	1,801.96	4,581.00	4,837.80
		100	421.79	422.36	1,711.44	1,728.29	5,777.45	5,678.51
	400	3	441.07	442.01	1,928.87	1,948.83	4,968.50	5,031.05
		10	459.86	463.82	2,971.65	2,956.95	2,996.15	2,892.58
		100	484.24	487.10	3,147.70	3,164.31	2,505.60	2,369.15
60	200	3	994.88	999.09	1,196.22	1,200.62	13,226.43	13,479.38
		10	1,012.49	1,018.56	1,852.71	1,836.65	13,072.60	13,059.84
		100	1,044.51	1,050.06	1,854.31	1,846.06	14,036.33	13,790.60
	400	3	1,334.01	1,342.53	1,997.95	2,013.29	10,258.13	10,420.38
		10	1,348.19	1,373.07	2,389.56	2,377.26	9,676.28	9,558.20
		100	1,471.60	1,487.80	2,702.07	2,698.91	9,433.98	9,324.15
100	200	3	1,184.89	1,193.46	1,542.37	1,578.03	14,592.98	14,883.33
		10	1,254.88	1,268.18	2,169.92	2,168.66	13,863.38	14,688.41
		100	1,336.98	1,351.89	2,070.42	2,068.13	15,547.10	15,463.70
	400	3	1,623.81	1,649.47	2,585.43	2,591.33	8,961.93	9,340.71
		10	1,722.33	1,760.55	2,751.19	2,755.60	9,140.43	9,027.38
		100	1,730.86	1,769.52	2,822.94	2,822.98	7,268.48	7,344.16
average			1,158.36	1,172.33	2,133.41	2,136.27	10,227.22	10,306.46

Table I.12: Results for DBLF with Spaces – w/o Rotation

Instances			*∅ttd*		∅ time [s]		∅ iterations	
n	m	item types	best	avg.	best	avg.	best	avg.
20	200	3	408.72	408.90	452.12	458.25	7,743.55	7,606.63
		10	417.73	418.43	1,600.41	1,621.40	5,154.85	4,903.65
		100	422.22	423.99	581.24	601.45	6,076.55	6,145.43
	400	3	441.99	444.38	1,829.69	1,846.42	4,644.60	4,944.77
		10	463.15	468.81	2,930.18	2,950.96	2,825.00	2,779.67
		100	486.25	490.85	3,097.23	3,107.85	2,595.85	2,640.54
60	200	3	1,007.42	1,024.58	788.50	785.25	14,231.63	14,045.66
		10	1,034.21	1,065.49	947.70	955.44	14,259.68	13,465.95
		100	1,072.97	1,107.22	794.92	803.29	14,048.58	14,293.42
	400	3	1,367.66	1,389.25	1,385.63	1,402.57	10,867.20	10,759.32
		10	1,391.70	1,455.47	1,177.77	1,180.26	9,961.60	9,745.82
		100	1,487.46	1,575.47	1,169.42	1,177.72	10,185.63	9,462.70
100	200	3	1,195.44	1,204.05	1,437.06	1,446.24	15,351.18	15,266.00
		10	1,260.62	1,273.53	1,943.74	1,957.20	15,131.15	14,856.15
		100	1,343.13	1,357.84	1,864.35	1,874.03	15,846.53	15,761.38
	400	3	1,648.48	1,690.96	1,799.21	1,798.69	10,590.55	10,681.94
		10	1,736.23	1,803.30	1,901.75	1,906.36	10,466.95	10,608.23
		100	1,742.13	1,823.30	2,005.08	2,015.79	8,572.33	8,650.50
average			1,173.83	1,206.54	1,497.37	1,506.40	10,935.55	10,807.16

Table I.13: Results for DBLF with Spaces – w/o LIFO

Instances			*∅ttd*		∅ time [s]		∅ iterations	
n	m	item types	best	avg.	best	avg.	best	avg.
20	200	3	408.21	408.48	329.31	330.97	7,506.25	7,648.59
		10	413.95	414.63	856.68	870.78	5,099.60	5,100.00
		100	419.98	420.81	535.98	545.40	5,097.40	5,247.48
	400	3	436.58	436.84	1,233.23	1,246.37	5,269.55	5,371.57
		10	444.93	447.24	2,437.06	2,444.79	2,970.10	3,041.40
		100	461.97	466.56	2,548.12	2,563.00	2,606.25	2,656.50
60	200	3	988.53	996.97	575.95	588.24	13,838.25	13,490.59
		10	972.46	1,004.84	807.19	813.71	13,136.53	13,296.18
		100	1,001.45	1,035.51	598.99	614.22	13,959.30	13,516.68
	400	3	1,320.57	1,330.34	1,206.76	1,234.29	10,539.65	10,470.26
		10	1,285.76	1,332.08	1,366.33	1,367.25	10,105.55	10,155.68
		100	1,388.06	1,453.33	1,139.85	1,136.29	10,238.88	9,819.24
100	200	3	1,173.70	1,180.70	1,259.46	1,269.14	14,282.20	14,273.39
		10	1,207.79	1,218.37	1,678.45	1,687.20	15,030.15	14,455.12
		100	1,277.65	1,292.84	1,539.75	1,551.51	16,147.00	15,636.21
	400	3	1,593.60	1,629.46	1,595.55	1,603.93	10,873.03	10,807.45
		10	1,642.82	1,707.21	1,783.92	1,792.05	10,738.83	10,623.41
		100	1,618.75	1,696.49	2,108.03	2,108.13	8,539.70	8,479.40
average			1,117.60	1,145.03	1,308.70	1,317.78	10,780.24	10,637.09

Table I.14: Results for DBLF with Spaces – w/o Minimal Supporting Area

Instances			*∅ttd*		∅ time [s]		∅ iterations	
n	m	item types	best	avg.	best	avg.	best	avg.
20	200	3	408.65	409.39	437.47	441.51	7,529.15	7,557.73
		10	416.33	417.51	1,520.85	1,559.53	5,134.50	4,970.59
		100	420.34	421.34	228.30	245.78	5,689.30	5,730.79
	400	3	440.58	441.25	2,002.58	2,012.66	3,926.50	3,949.37
		10	458.05	462.06	2,879.00	2,901.52	3,108.95	2,831.01
		100	471.97	475.01	2,601.71	2,608.84	2,653.40	2,489.50
60	200	3	995.29	1,008.50	819.68	824.65	14,436.88	14,013.88
		10	989.98	1,016.94	967.64	972.97	12,824.20	12,990.43
		100	1,010.01	1,035.21	721.39	730.13	14,302.30	13,939.84
	400	3	1,322.57	1,345.48	1,481.23	1,506.78	10,727.40	10,417.03
		10	1,302.24	1,350.97	1,342.05	1,355.23	10,526.20	10,382.11
		100	1,378.11	1,433.78	1,161.19	1,164.13	10,451.55	10,227.31
100	200	3	1,177.41	1,186.67	1,513.40	1,512.08	15,165.43	14,610.00
		10	1,226.13	1,236.55	1,902.86	1,913.58	14,862.10	14,648.62
		100	1,287.70	1,301.14	1,691.43	1,690.43	15,522.55	15,801.54
	400	3	1,605.99	1,644.84	1,813.54	1,824.48	10,526.28	10,321.45
		10	1,663.03	1,716.63	1,921.66	1,924.27	10,701.13	10,610.26
		100	1,633.08	1,696.28	2,084.52	2,101.14	8,536.35	8,554.64
average			1,126.63	1,152.42	1,483.70	1,493.65	10,840.22	10,685.44

Table I.15: Results for DBLF with Spaces – w/o Fragility

Instances			*∅ttd*		∅ time [s]		∅ iterations	
n	m	item types	best	avg.	best	avg.	best	avg.
20	200	3	408.56	410.11	419.81	422.88	7,293.25	7,758.01
		10	416.89	417.44	1,499.46	1,524.56	4,809.90	5,010.31
		100	421.21	422.61	269.33	285.52	5,968.30	5,881.23
	400	3	440.05	441.23	1,751.98	1,765.15	5,387.20	5,341.13
		10	459.32	464.01	2,806.26	2,817.79	2,973.35	2,989.39
		100	476.45	480.99	2,918.19	2,929.88	3,014.30	2,787.38
60	200	3	1,000.50	1,016.45	715.47	724.41	14,121.63	14,104.52
		10	1,011.92	1,042.87	844.95	844.39	13,806.13	13,398.73
		100	1,046.86	1,076.19	696.98	695.86	14,420.10	14,311.00
	400	3	1,327.26	1,348.91	1,454.52	1,453.03	10,668.05	10,753.20
		10	1,334.51	1,394.52	1,207.79	1,213.79	9,325.65	9,257.27
		100	1,438.99	1,517.72	1,096.30	1,099.78	9,474.13	9,479.32
100	200	3	1,181.27	1,191.09	1,375.64	1,377.47	14,913.50	14,943.72
		10	1,245.55	1,256.90	1,920.51	1,931.45	15,433.88	14,995.97
		100	1,318.60	1,332.38	1,775.83	1,778.25	16,679.58	16,142.49
	400	3	1,614.25	1,658.45	1,761.90	1,769.93	11,119.53	10,730.44
		10	1,691.75	1,757.89	1,948.17	1,942.49	10,705.50	10,627.61
		100	1,685.57	1,761.36	2,037.85	2,050.86	8,690.75	8,650.82
average			1,147.22	1,178.19	1,444.56	1,450.31	10,938.77	10,818.59

Table I.16: Results for DBLF with Spaces – w/o Load Capacity

Instances			*∅ttd*		∅ time [s]		∅ iterations	
n	m	item types	best	avg.	best	avg.	best	avg.
20	200	3	406.08	409.12	446.32	446.63	7,412.45	7,401.08
		10	416.80	417.77	1,449.79	1,450.19	5,280.60	5,038.72
		100	422.40	423.89	1,299.14	1,300.57	5,947.80	6,019.43
	400	3	437.21	440.89	1,946.63	1,944.76	3,811.80	4,061.39
		10	460.34	465.80	2,667.39	2,672.47	3,143.75	3,048.27
		100	484.64	488.58	3,118.45	3,121.29	2,711.60	2,776.69
60	200	3	993.82	996.09	919.31	923.67	14,213.73	13,495.38
		10	1,010.14	1,018.80	1,366.22	1,368.87	13,807.43	13,671.16
		100	1,043.73	1,053.38	1,292.82	1,293.85	14,825.48	14,343.13
	400	3	1,331.68	1,338.69	1,870.96	1,874.00	9,689.85	9,531.50
		10	1,350.31	1,363.80	2,081.61	2,101.58	10,670.30	10,801.72
		100	1,473.23	1,489.41	2,398.97	2,394.19	11,181.73	10,997.51
100	200	3	1,173.16	1,179.29	1,170.69	1,181.61	14,161.98	14,151.12
		10	1,242.86	1,253.01	1,539.73	1,547.17	15,963.08	15,836.05
		100	1,332.48	1,341.65	1,722.82	1,714.17	15,706.48	15,489.47
	400	3	1,621.09	1,638.17	2,550.46	2,551.55	8,972.85	8,454.31
		10	1,720.00	1,747.77	2,647.00	2,642.66	8,751.63	8,474.87
		100	1,729.44	1,763.16	2,610.94	2,607.27	7,972.75	8,259.76
average			1,155.71	1,167.08	1,842.36	1,844.57	10,671.42	10,511.92

Table I.17: Results for DBLF with RTree – All Constraints

Instances			∅*ttd*		∅ time [s]		∅ iterations	
n	m	item types	best	avg.	best	avg.	best	avg.
20	200	3	408.91	411.15	1,171.58	1,175.36	6,049.70	6,203.83
		10	418.90	422.45	1,766.07	1,769.97	4,227.85	4,281.71
		100	423.33	427.00	2,305.45	2,313.50	4,182.85	4,176.75
	400	3	443.15	449.27	2,482.88	2,517.98	3,688.00	3,446.39
		10	465.89	477.33	2,604.76	2,604.60	2,980.70	2,865.67
		100	492.93	506.85	2,492.67	2,498.13	3,202.80	3,116.92
60	200	3	1,036.32	1,050.28	2,931.78	2,943.80	3,951.33	3,852.55
		10	1,102.85	1,131.68	2,992.54	2,997.29	2,825.28	2,750.34
		100	1,165.00	1,190.18	3,301.62	3,307.65	2,180.15	2,315.36
	400	3	1,391.19	1,417.44	2,856.99	2,870.60	3,980.13	4,199.43
		10	1,465.84	1,522.38	2,779.39	2,792.66	5,581.85	5,761.70
		100	1,607.81	1,680.72	2,304.35	2,318.99	6,661.90	6,834.69
100	200	3	1,242.08	1,259.40	3,282.33	3,289.11	3,686.55	3,568.99
		10	1,394.36	1,405.74	3,517.30	3,521.70	2,220.23	2,103.43
		100	1,512.91	1,532.77	3,485.58	3,487.72	1,404.75	1,409.09
	400	3	1,739.93	1,759.22	3,389.67	3,395.15	2,872.33	2,576.32
		10	1,950.96	1,982.12	3,600.00	3,600.00	422.03	422.33
		100	2,012.89	2,050.93	3,600.00	3,600.00	227.48	223.41
average			1,263.25	1,288.66	2,963.55	2,970.96	3,212.00	3,204.22

Table I.18: Results for DBLF with RTree – w/o Rotation

Instances			∅*ttd*		∅ time [s]		∅ iterations	
n	m	item types	best	avg.	best	avg.	best	avg.
20	200	3	408.85	409.70	1,205.88	1,215.16	5,159.90	5,366.50
		10	420.34	424.29	1,811.27	1,811.84	4,127.05	4,159.02
		100	424.16	428.91	1,787.82	1,796.72	3,611.10	3,749.56
	400	3	446.10	453.04	2,584.06	2,602.60	1,500.95	1,542.49
		10	472.71	480.51	2,632.29	2,642.64	989.45	987.88
		100	494.58	507.82	2,567.03	2,560.71	713.15	712.91
60	200	3	1,043.12	1,063.39	1,854.86	1,860.17	3,851.10	3,898.28
		10	1,115.25	1,143.96	1,949.03	1,953.80	3,019.58	2,863.58
		100	1,165.34	1,198.22	1,950.89	1,952.51	2,656.65	2,661.63
	400	3	1,425.22	1,451.14	2,953.36	2,969.49	2,194.63	1,701.78
		10	1,497.34	1,554.39	2,818.67	2,825.45	4,175.38	4,406.43
		100	1,613.93	1,696.99	2,310.29	2,313.46	6,270.70	6,026.75
100	200	3	1,252.47	1,280.29	3,356.19	3,360.01	110.38	111.84
		10	1,386.87	1,414.46	3,549.66	3,551.13	92.85	93.56
		100	1,517.46	1,543.80	3,520.80	3,521.71	98.00	97.92
	400	3	1,766.55	1,789.38	3,332.02	3,333.50	3,171.93	2,753.72
		10	1,976.12	2,003.44	3,600.00	3,600.00	451.03	457.74
		100	2,025.83	2,071.39	3,600.00	3,600.00	256.63	257.68
average			1,274.59	1,304.20	2,739.33	2,743.74	2,293.31	2,239.34

Table I.19: Results for DBLF with RTree – w/o LIFO

Instances			∅*ttd*		∅ time [s]		∅ iterations	
n	m	item types	best	avg.	best	avg.	best	avg.
20	200	3	408.38	408.61	984.83	992.25	6,169.15	6,239.68
		10	414.69	415.68	1,782.06	1,805.08	4,201.20	4,231.15
		100	420.93	422.44	1,367.34	1,371.89	4,269.55	4,315.68
	400	3	436.90	439.20	2,235.03	2,256.21	2,520.10	2,328.57
		10	447.88	452.47	2,608.75	2,611.99	1,831.70	1,922.01
		100	469.65	475.92	2,547.27	2,561.16	2,127.85	2,165.85
60	200	3	1,012.61	1,023.80	1,759.85	1,777.77	4,051.08	3,959.32
		10	1,036.71	1,062.54	1,931.42	1,938.81	2,801.40	2,678.86
		100	1,077.29	1,106.44	1,993.17	1,993.77	2,785.73	2,627.81
	400	3	1,355.40	1,375.43	2,968.12	2,972.43	1,415.23	1,319.17
		10	1,390.20	1,430.49	2,923.16	2,930.62	2,244.55	2,048.95
		100	1,509.19	1,570.97	2,441.35	2,437.34	3,951.83	3,761.41
100	200	3	1,213.02	1,236.55	3,358.97	3,361.01	124.68	124.42
		10	1,296.55	1,319.25	3,474.56	3,477.12	98.43	161.79
		100	1,385.49	1,412.18	3,519.67	3,520.48	107.60	106.71
	400	3	1,658.49	1,694.96	3,303.74	3,315.40	2,363.25	2,318.42
		10	1,807.75	1,843.33	3,446.43	3,452.90	574.85	569.48
		100	1,839.21	1,875.84	3,600.00	3,600.00	286.55	289.99
average			1,192.07	1,217.26	2,698.87	2,705.13	2,091.00	2,037.85

Table I.20: Results for DBLF with RTree – w/o Minimal Supporting Area

Instances			∅*ttd*		∅ time [s]		∅ iterations	
n	m	item types	best	avg.	best	avg.	best	avg.
20	200	3	409.02	411.93	1,217.62	1,221.52	5,653.15	5,459.69
		10	418.78	422.33	1,811.86	1,812.65	4,151.65	4,140.28
		100	422.05	424.68	1,662.07	1,679.56	3,950.30	3,873.98
	400	3	442.26	448.75	2,600.08	2,592.97	1,500.20	1,493.55
		10	465.88	473.27	2,633.38	2,638.83	1,075.75	1,139.83
		100	478.64	484.34	2,565.56	2,570.81	1,312.30	1,230.11
60	200	3	1,024.66	1,046.45	1,937.71	1,938.96	3,568.85	3,440.33
		10	1,067.60	1,098.87	1,972.29	1,974.16	2,572.03	2,508.18
		100	1,106.67	1,135.34	2,073.62	2,073.06	2,520.93	2,513.09
	400	3	1,389.93	1,413.70	2,954.96	2,972.97	1,064.75	1,243.63
		10	1,422.27	1,465.33	2,998.13	2,996.04	2,555.25	2,676.65
		100	1,512.42	1,571.53	2,440.92	2,439.39	3,224.65	3,796.01
100	200	3	1,241.36	1,271.63	3,355.11	3,358.37	109.15	107.84
		10	1,355.34	1,377.03	3,550.76	3,551.14	93.03	91.66
		100	1,449.49	1,472.96	3,522.94	3,519.58	100.75	98.55
	400	3	1,721.49	1,754.70	3,402.28	3,407.15	1,672.33	1,651.29
		10	1,874.06	1,912.97	3,600.00	3,600.00	310.43	308.31
		100	1,904.12	1,932.25	3,600.00	3,600.00	161.18	159.53
average			1,225.85	1,252.36	2,776.93	2,779.27	1,785.00	1,817.59

Table I.21: Results for DBLF with RTree – w/o Fragility

Instances			∅*ttd*		∅ time [s]		∅ iterations	
n	m	item types	best	avg.	best	avg.	best	avg.
20	200	3	408.95	411.27	1,212.20	1,218.75	5,699.95	5,795.96
		10	418.33	421.58	1,808.73	1,809.52	4,195.30	4,171.03
		100	422.74	425.69	1,605.23	1,612.80	4,070.10	4,202.12
	400	3	442.81	447.28	2,595.09	2,601.50	1,530.70	1,535.40
		10	465.30	474.08	2,632.39	2,638.59	1,168.60	1,067.19
		100	486.02	493.46	2,550.61	2,560.04	1,125.20	1,131.14
60	200	3	1,031.36	1,055.23	1,920.20	1,930.07	4,056.83	3,880.08
		10	1,088.32	1,118.37	1,953.63	1,958.13	2,963.55	2,899.51
		100	1,140.98	1,169.42	2,039.41	2,041.04	2,698.05	2,661.66
	400	3	1,385.14	1,412.25	2,958.28	2,963.86	1,356.23	1,523.61
		10	1,446.68	1,505.91	2,938.83	2,927.42	4,452.03	3,824.71
		100	1,568.61	1,640.17	2,424.07	2,424.59	5,741.78	5,304.28
100	200	3	1,243.45	1,270.47	3,354.67	3,362.08	131.60	133.46
		10	1,379.64	1,398.32	3,547.80	3,551.19	96.50	105.78
		100	1,479.96	1,503.80	3,521.26	3,521.37	167.65	119.22
	400	3	1,735.86	1,766.28	3,366.69	3,367.99	2,875.40	2,627.00
		10	1,910.17	1,946.92	3,600.00	3,600.00	451.20	457.35
		100	1,964.84	1,999.43	3,600.00	3,600.00	253.68	259.18
average			1,246.47	1,274.88	2,761.80	2,764.56	2,275.96	2,183.15

Table I.22: Results for DBLF with RTree – w/o Load Capacity

Instances			∅*ttd*		∅ time [s]		∅ iterations	
n	m	item types	best	avg.	best	avg.	best	avg.
20	200	3	407.17	410.38	1,122.76	1,123.53	6,353.60	5,735.38
		10	419.02	421.77	1,621.74	1,621.82	4,372.05	4,417.89
		100	424.27	426.92	2,016.84	2,018.54	4,467.60	4,506.84
	400	3	440.77	446.19	2,369.67	2,375.68	2,166.85	2,580.01
		10	466.56	474.48	2,632.50	2,639.59	1,942.75	1,990.15
		100	491.67	501.74	2,558.14	2,567.58	1,895.55	1,987.62
60	200	3	1,034.36	1,045.17	2,812.16	2,809.04	3,131.78	3,229.33
		10	1,095.67	1,115.99	2,816.16	2,816.85	2,660.38	2,571.07
		100	1,152.61	1,174.51	2,790.89	2,795.01	2,320.85	2,342.47
	400	3	1,392.88	1,413.55	2,572.47	2,580.20	2,182.58	2,257.93
		10	1,467.76	1,493.34	2,385.76	2,400.64	1,639.05	1,688.41
		100	1,606.68	1,629.68	2,144.16	2,153.80	944.90	925.72
100	200	3	1,229.01	1,244.98	2,845.52	2,844.13	4,037.95	3,899.78
		10	1,384.05	1,396.98	2,984.17	2,983.59	2,524.25	2,439.00
		100	1,495.08	1,516.26	3,247.58	3,248.27	1,698.25	1,704.71
	400	3	1,739.13	1,761.02	3,111.58	3,113.11	3,029.63	2,693.61
		10	1,952.74	1,981.10	3,589.12	3,587.19	603.68	601.40
		100	2,013.95	2,044.17	3,600.00	3,600.00	319.30	317.77
average			1,259.24	1,277.17	2,737.36	2,740.35	2,379.45	2,352.01

Table I.23: Results for Zhang et al. (2017) Instances

Algorithm	constraint set	*Øttd*		∅ time [s]		∅ iterations	
		best	avg.	best	avg.	best	avg.
DBLF with Points	All Constraints	784.50	788.53	500.52	503.75	17,160.67	17,749.56
	w/o Rotation	790.17	797.09	299.02	297.49	17,417.59	17,468.40
	w/o LIFO	728.22	733.14	294.48	293.31	16,512.89	15,740.81
	w/o Minimal S. Area	741.43	746.89	281.62	282.15	18,127.56	17,035.43
	w/o Fragility	769.22	776.54	274.02	278.05	17,087.70	17,037.24
	w/o Load Capacity	746.34	750.87	284.17	281.21	17,271.48	17,488.07
average		**759.98**	**765.51**	**322.30**	**322.66**	**17,262.98**	**17,086.59**
DBLF with Spaces	All Constraints	778.19	782.43	476.40	467.67	17,276.37	17,338.45
	w/o Rotation	786.36	791.10	255.27	253.55	17,969.11	17,031.30
	w/o LIFO	723.02	728.62	294.41	284.69	17,459.63	16,342.47
	w/o Minimal S. Area	734.72	739.33	247.04	244.50	16,645.52	16,373.58
	w/o Fragility	763.95	766.22	268.38	271.03	17,333.59	16,733.05
	w/o Load Capacity	738.42	743.78	288.18	292.40	17,615.37	17,119.99
average		**754.11**	**758.58**	**304.95**	**302.31**	**17,383.27**	**16,823.14**
DBLF with RTree	All Constraints	874.95	874.91	1,962.35	1,947.49	7,668.78	7,772.15
	w/o Rotation	869.32	880.53	2,077.42	2,080.08	7,960.70	7,454.01
	w/o LIFO	785.39	797.30	2,067.01	2,074.40	7,227.78	6,929.02
	w/o Minimal S. Area	823.46	841.06	2,079.90	2,085.14	7,567.52	7,244.15
	w/o Fragility	852.32	857.95	2,066.63	2,076.38	8,095.26	7,945.50
	w/o Load Capacity	826.78	840.67	2,123.77	2,122.91	5,423.78	5,276.93
average		**838.70**	**848.74**	**2,062.85**	**2,064.40**	**7,323.97**	**7,103.63**

Paper II

Axle Weights in combined Vehicle Routing and Container Loading Problems

Corinna Krebs, Jan Fabian Ehmke

Published in *EURO Journal on Transportation and Logistics*, June 2021, volume 10, pp. 123–456.
DOI: 10.1016/j.ejtl.2021.100043.

II

Abstract

Overloaded axles not only lead to increased erosion on the road surface, but also to an increased braking distance and more serious accidents due to higher impact energy. Therefore, the load on axles should be already considered during the planning phase and thus before loading the truck in order to prevent overloading. Hereby, a detailed 2D or 3D planning of the vehicle cargo space is required. We model the Axle Weight Constraint for trucks with and without trailers based on the Science of Statics and provide flexible formulas for different axle configurations of trucks. We include the Axle Weight Constraint into the combined Vehicle Routing and Container Loading Problem (“2L-CVRP" and “3L-CVRP"). A hybrid heuristic approach is used where an outer Adaptive Large Neighbourhood Search tackles the routing problem and an inner Deepest-Bottom-Left-Fill algorithm solves the packing problem. Moreover, to ensure feasibility, we show that the Axle Weight Constraint must be checked after each placement of an item. The impact of the Axle Weight Constraint is also evaluated.

Contents

1 Introduction

A survey by Blower and Woodrooffe (2012) of the University of Michigan analysed the status of truck safety for four countries: Australia, Brazil, China and the United States. The overloading of trucks is a major problem especially for China and Brazil. According to reports in China, 70–90% of truck accidents are related to overloaded and oversized trucks. In Brazil, 60% of the trucks in crashes have been revealed to be overloaded and 20% of registered trucks exceeding the gross mass. The overloading of axles can result in an overheating and failing of tyres and brakes. If the permissible axle weight of a steering axle is exceeded, the steering is more cumbersome and it is possible to loose control of the vehicle. There are also economic effects: A study by Pais et al. (2013) showed that the increased erosion of roads raises the pavement costs by more than 100%. For these reasons, it is essential to consider the load on the axles of trucks directly in the planning of routes.

In the majority of previous research on the Vehicle Routing Problem (VRP), customer demand was simply expressed as the total mass or volume. This is not sufficient to take axle weights into account, as a detailed 2D or 3D planning of the vehicle cargo space is required. In this paper, we present simple, yet effective formulas for considering the axle weights suitable for all 2D and 3D Container Loading Problems. We include our approach in the combinations of Vehicle Routing and Container Loading Problems, such as the 2L-CVRP and 3L-CVRP. Moreover, we propose 80 new instances dealing with Semi-Truck Trailers varying systematically in the number of customers, item types and items. To ensure transparency, we have published online all results in detail including the packing plans with the exact positions of the items.

This paper considers two aspects for the first time: I) We formulate a flexible, adjustable approach to consider axle weights for trucks with or without trailers for all axles and II) we include the Axle Weight Constraint in the 3L-CVRP. Moreover, we show with a detailed real example that it is necessary to check the Axle Weight Constraint after each placement of an item, since the mass of items can act on an axle, but it can also relieve an axle. Therefore, also unloading an item can lead to an overloading of one or several axles.

We use a hybrid heuristic algorithm for tackling the Vehicle Routing and the Container Loading Problem. The routing heuristic is based on the Adaptive Large Neighbourhood Search (ALNS) by Koch et al. (2018) calling for each route a modified packing heuristic based on the Deepest-Bottom-Left-Fill (DBLF) algorithm proposed by Karabulut and İnceoğlu (2005). Both heuristics are described in detail in the following subsections. For the computational tests, we use a new instance set and well-known instances from the literature.

The related literature is reviewed in Section 2. The considered problems (2L- and 3L-CVRP) are formulated in Section 3. In Section 4, the Axle Weight Constraint is described in detail. The hybrid solution approach is explained in Section 5. Section 6 presents computational results, analysing the impact of the Axle Weight Constraint on VRP solutions. Finally, conclusions are drawn in Section 7.

2 Literature Review

In this section, the literature considering the Axle Weight Constraint with their vehicle models is reviewed. To the best of our knowledge, the following overview represents all papers considering axle weights. First, the papers dealing with the Container Loading Problem or a variant of this

problem are regarded since this represents a subproblem of the problems considered in this paper. Then, papers are summarized that include the Axle Weight Constraint into the combined Container Loading and Vehicle Routing Problem.

2.1 Container Loading Problem

Although Container Loading Problems have been investigated for several decades, the axle weights of vehicles have so far only been considered in the following three papers. In the 3D Single Container Loading Problem, a number of three-dimensional boxes must be packed into one three-dimensional container while minimizing the total volume utilization. In this context, the Axle Weight Constraint was first considered in a paper by Lim et al. (2013). They examine the general rules of the California Vehicle Code. They simplify the rules by formulating three semi-trailer truck models with three limits each: One limit for the front axle group, one for the rear axle group and one for the gross vehicle weight. Then, a Greedy Randomized Adaptive Search Procedure Wall-Building Algorithm packs the boxes, and the Axle Weight Constraint is checked. For this, the ideal mass center of the container is calculated. Then, the deviation between the ideal mass center and the current mass center of the container is minimized by rearranging the walls of loaded boxes.
The Multi Container Loading Problem is an extension of the Single Container Loading Problem, where a set of items needs to be packed into multiple containers while minimizing the number of used containers. For the 2D problem variant, the Axle Weight Constraint is included in mathematical models by Alonso et al. (2017). The Axle Weight Constraint is considered for trucks with two axles and is included in the model by formulating the equilibrium for the moments and forces. In Alonso et al. (2019), this model is extended by further constraints such as dynamic stability constraints.

2.2 Combined Vehicle Routing and Container Loading Problem

The VRP is one of the most studied optimization problems in logistics. This problem involves the optimal planning of routes to deliver goods to customers that are located in a depot. This is accomplished by a fleet of vehicles having a certain capacity. Since a detailed 2D and 3D planning of the cargo space is required to take axle weights into account, the Axle Weight Constraint has rarely been taken into account so far. In Iori et al. (2007), the combination of the 2D Container Loading Problem with the Capacitated Vehicle Routing Problem (2L-CVRP) is introduced. The objective is to minimize the total travel cost (distance). The Axle Weight Constraint was first considered in the 2L-CVRP by Pollaris et al. (2016). The problem is solved by a mixed integer linear programming formulation. In their model, a truck with a trailer is considered. The axle weights are calculated only for the trailer axle group and the driving axle. The separated examination of axle weights of the steering and driving axle of the tractor is not considered. The axle weights are calculated by means of equilibrium of forces and moments. Since in their model the pallets are packed alternately in two rows, the formulas for calculating the axle loads are fixed to these positions. Pollaris et al. (2016) proposed 128 instances varying in the number of customers, the number of pallets per customer and the masses of the pallets. The instances are tested with CPLEX, but for some instances, no feasible solution can be found after two hours of runtime. To solve larger instances, in Pollaris et al. (2017), an Iterated Local Search approach with Sequence-Based Pallet Loading is developed. In this framework, the same assumptions are used for the axle weight calculation and the vehicle model. Additional 96 instances were created with up to 100 customers, which can be solved by the proposed metaheuristic.
The Three-Dimensional Loading Capacitated Vehicle Routing Problem (3L-CVRP) is an extension of the 2L-CVRP combining 3D Container Loading with the Capacitated Vehicle Routing Problem introduced by Gendreau et al. (2006). The 3L-CVRP has been studied intensively in the recent years so that the results for this benchmark have been improved repeatedly by researchers (e.g. Tarantilis et al. (2009), Fuellerer et al. (2010), Bortfeldt (2012) and Wei et al. (2014)). However, the Axle Weight Constraint has not been included in the 3L-CVRP yet. To the best of our knowledge, this paper considers for the first time the Axle Weight Constraint in the 3L-CVRP. Moreover, we provide a flexible approach, where the feasibility of an arbitrary position of an item can be checked with respect to axle weights. This approach is adaptable to various vehicle types.

3 Problem Formulation

To model the 2L-CVRP and 3L-CVRP, we follow the convention by Bortfeldt (2012). 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. Let the euclidean distance $co_{i,j}$ ($co_{i,j} > 0$) be assigned to each edge $e_{i,j} \in E$ ($i \neq j, i, j = 0, ..., n$). The demand of customer $i \in N \setminus \{0\}$ consists of c_i items. 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 total demanded mass for a customer i is given by mc_i. The items are delivered by at most v_{max} available, homogenous and capacitated vehicles. Each vehicle has a maximum load capacity D and a cuboid cargo space defined by length L, width W and height H.
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 cargo space for each item included in the route.

A solution is feasible if

(S1) All routes R_v and packing plans PP_v ($v = 1, ..., v_{used}$) are feasible (see below);

(S2) The number of used vehicles v_{used} does not exceed the number of available vehicles v_{max};

(S3) 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) Each customer is visited exactly once.

Each packing plan must obey the following loading constraint set P:

(C1) *Geometry*: The items must be packed without overlapping and within the vehicle (e.g. obeying the maximum length, width and height of the vehicle);

(C2) *Orthogonality*: The items can only be placed orthogonally inside a vehicle;

(C3) *Load Capacity*: The sum of masses of all included items of a vehicle does not exceed the maximum load capacity D;

(C4) *LIFO*: No item is placed above or in front of item $I_{i,k}$, which belongs to a customer served after customer i.

For the 3L-CVRP, the following loading constraints must be additionally respected:

(C5) *Rotation*: The items can be rotated 90° only on the width-length plane;

(C6) *Minimal Supporting Area*: Each item has a supporting area of at least a percentage α of its base area;

(C7) *Fragility*: No non-fragile items are placed on top of fragile items.

Moreover, the Axle Weight Constraint is considered.

(C8) *Axle Weight*: The load on the axles do not exceed the permissible axle weights.

The 2L-CVRP and 3L-CVRP consist of determining a feasible solution minimizing the total travel distance (ttd) and obeying all above constraints.

4 Axle Weight Constraint

This section details our approaches for the consideration of axle weights in the problem at hand. Based on the Science of Statics, the formulas for calculating the axle weights for a box truck are derived. Then, resultant axles for replacing axle groups are introduced. Afterwards, the formulas for calculating the axle weights for trucks with trailers are shown. Then, we demonstrate that the Axle Weight Constraint must be checked after each placement of an item. The following formulas are used to check if the position $(x_{i,k}, y_{i,k}, z_{i,k})$ of an item $I_{i,k}$ is feasible according to the Axle Weight Constraint. If an axle weight limit is exceeded, another position for the item must be found. This flexible approach makes it easy to check the feasibility of an item position.

4.1 Approach for Box Trucks

The following formulas are suitable for the axle weight calculation of box trucks and box vans. These are vehicles with a cuboid-shaped cargo area without any kind of trailer. For implementation, additional specifications of the vehicle are needed. In practice, those are given by the manufacturers. 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 the unit of mass. The parameter L_f describes the distance between the front axle and the cargo area. The wheelbase WB is the distance between the front and the rear axle (see Fig. II.1).

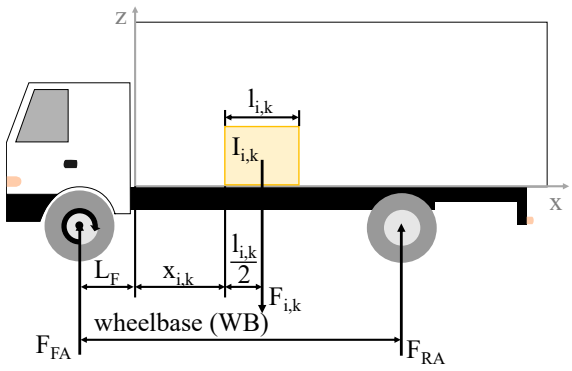


Figure II.1: Vehicle Data

An object on the surface of the earth experiences a force caused by the gravitational attraction of the earth. This force is calculated as the mass of the object times the acceleration of gravity g ($g \approx 9.81 \frac{m}{s^2}$). Therefore, each item $I_{i,k}$ acts a force $F_{i,k}$ on the vehicle, which is

$$F_{i,k} = m_{i,k} \cdot g. \tag{II.1}$$

The point of the force $F_{i,k}$ is located in the center of mass for each item. If the mass of an item is homogeneously distributed, the center of mass and the geometric center are at the same point. Each axle counteracts these forces, whereby F_{FA} is the force representing the front axle and F_{RA} the force for the rear axle.
Moreover, each force creates a moment, which can be determined to any point in the system. It is expedient to determine the moments to an unknown force, such as the force of the front axle F_{FA}. Each moment is calculated by the force multiplied by the distance r from the front axle to the force. Thus, each item's force creates a moment $M_{i,k}$. Supposing that the mass of an item is homogeneously distributed, resulting that the point of force lays in the geometric center, the distance $r_{i,k}$ to the point of the front axle force F_{FA} is

$$r_{i,k} = L_f + x_{i,k} + l_{i,k}/2. \tag{II.2}$$

The moment $M_{i,k}$ created by item $I_{i,k}$ is:

$$M_{i,k} = F_{i,k} \cdot r_{i,k} \tag{II.3}$$

or rather

$$M_{i,k} = F_{i,k} \cdot (L_f + x_{i,k} + l_{i,k}/2). \tag{II.3b}$$

When rotating an item, $l_{i,k}$ and $w_{i,k}$ are swapped. The force from the rear axle F_{RA} creates another moment, which is F_{RA} multiplied by the wheelbase WB.
In the Science of Statics, the forces and moments are in static equilibrium with their environment. Thus, the summation of forces F and of moments M are zero. Considering the direction of the forces and moments, the following formulas can be applied for a vehicle v.

Equilibrium of forces:

$$\sum_{i=1|i\in R_v}^{n} \sum_{k=1}^{c_i} F_{i,k} - F_{RA} - F_{FA} = 0, \tag{II.4}$$

which can be transformed to F_{FA}:

$$F_{FA} = \sum_{i=1|i\in R_v}^{n} \sum_{k=1}^{c_i} F_{i,k} - F_{RA}. \tag{II.4b}$$

The summation of moments must be zero and is:

$$\sum_{i=1|i\in R_v}^{n} \sum_{k=1}^{c_i} M_{i,k} - F_{RA} \cdot WB = 0, \tag{II.5}$$

which can be transformed to F_{RA}:

$$F_{RA} = \frac{1}{WB} \cdot (\sum_{i=1|i\in R_v}^{n} \sum_{k=1}^{c_i} M_{i,k}). \tag{II.5b}$$

As the position $(x_{i,k}, y_{i,k}, z_{i,k})$ of an item $I_{i,k}$ is checked w.r.t. feasibility, all values are known in Equation II.5b and the result for F_{RA} can be calculated and inserted in Equation II.4b to receive F_{FA}. The acting forces for the front and the rear axles must be below the permissible ones:

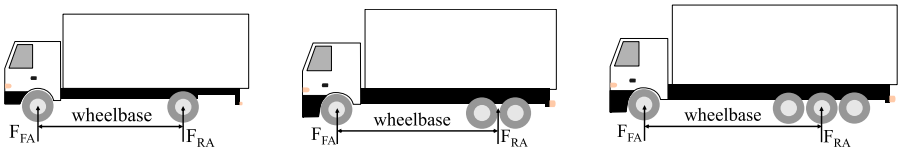
$$F_{FA} \leq FA_{perm} \cdot g \tag{II.6}$$

and

$$F_{RA} \leq RA_{perm} \cdot g. \tag{II.7}$$

4.2 Axle Group and Resultant Axle

The formulas shown in Section 4.1 are suitable for trucks with one front and rear axle each. Trucks as well as trailers can have axle groups consisting of two consecutive axles ("tandem-axle") or three consecutive axles ("tridem-axle"). For these axle groups, a so-called "resultant axle" replaces the axle group. The resultant axle is located in the center of the axle group and its value is the sum of each consecutive axle. Thus, as shown in Fig II.2, for a tandem-axle, the resultant axle is between the two rear axles. For a tridem-axle, the resultant axle lays in the middle axle of the three axles.



(a) One Axle (b) Tandem-Axle (c) Tridem-Axle

Figure II.2: Examples for Resultant Axles replacing Axle Groups

4.3 Approach for Semi-Trailer Trucks

The calculation of the load on the axles shown in Section 4.1 is suitable for trucks without a trailer. Semi-trailer trucks, as shown in Fig. II.3, have at least three axles, for which the loads must be determined: The load on the front axle and the rear axle of the tractor unit (truck) and the load on the axle group of the trailer. Their permissible load on these axles are FA_{perm} for the front axle, RA_{perm} for the rear axle and TA_{perm} for the trailer axle. Due to the three axles, the above formulas must be adapted. The Science of Statics is also applied to the following formulas.

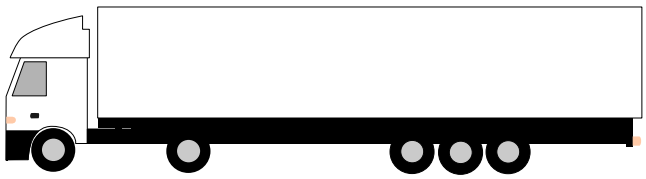
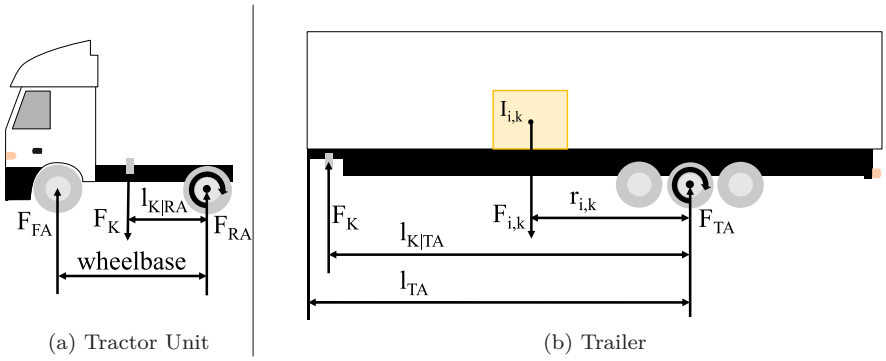


Figure II.3: Semi-Trailer Truck with Tridem Trailer Axle

Since the load on the axles of the tractor unit depends on the trailer, the forces and moments of the tractor unit and the trailer are examined separately. The kingpin on the semi-trailer connects the trailer with the tractor unit. In Fig.II.4 the distances, forces and moments are illustrated.



(a) Tractor Unit (b) Trailer

Figure II.4: Forces and Moments for Semi-Trailer Trucks

In the first step, the forces and moments of the trailer are determined. As shown in Section 4.2, a resultant trailer axle can be used instead of the tridem-axle. All moments of the trailer are calculated to the center of the resultant axle. Therefore, all distances from each force to this point must be determined. Let l_{TA} be the distance between the cargo area to the resultant trailer axle. The distance between the kingpin and the resultant axle of the trailer is denoted by $l_{K|TA}$. The distance $r_{i,k}$ between the item $I_{i,k}$ to the resultant trailer axle is

$$r_{i,k} = l_{TA} - x_{i,k} - \frac{l_{i,k}}{2}. \tag{II.8}$$

The force of the kingpin and the force of the resultant trailer axle act against the item forces. As demonstrated before, the summation of forces is zero in the Science of Statics. Therefore, the following must apply for the trailer:

$$F_{TA} + F_K - \sum_{i=1|i\in R_v}^{n} \sum_{k=1}^{c_i} F_{i,k} = 0. \tag{II.9}$$

This can be transformed to F_{TA}:

$$F_{TA} = \sum_{i=1|i\in R_v}^{n} \sum_{k=1}^{c_i} F_{i,k} - F_K. \tag{II.9b}$$

Similarly, the summation of the moments must be zero, so that it is

$$F_K \cdot l_{K|TA} - \sum_{i=1|i\in R_v}^{n} \sum_{k=1}^{c_i} F_{i,k} \cdot r_{i,k} = 0, \tag{II.10}$$

and transformed to F_K, one gets:

$$F_K = \frac{1}{l_{K|TA}} \cdot (\sum_{i=1|i\in R_v}^{n} \sum_{k=1}^{c_i} F_{i,k} \cdot r_{i,k}). \tag{II.10b}$$

Since all values in Equation II.10b are known, the force F_K can be calculated and inserted in Equation II.9b to receive F_{TA}. In the next step, the force and moment equilibriums are described for the tractor unit. The load on the tractor unit is carried by two axles. The front axle force F_{FA} and the rear axle force F_{RA} work against the force in the kingpin F_K. The summation of forces must be zero and is

$$F_{FA} + F_{RA} - F_K = 0. \tag{II.11}$$

This can be transformed to F_{RA}:

$$F_{RA} = F_K - F_{FA}. \tag{II.11b}$$

The moments are calculated in the center of the rear axle. Let WB be the wheelbase between the front and the rear axle of the tractor unit and $l_{K|RA}$ be the distance between the rear axle and the kingpin. Then, the summation of the following moments must be zero:

$$F_{FA} \cdot WB - F_K \cdot l_{K|RA} = 0. \tag{II.12}$$

The force F_K was calculated before. Thus, the force F_{FA} can be determined:

$$F_{FA} = \frac{1}{WB} \cdot F_K \cdot l_{K|RA}. \tag{II.12b}$$

The result for F_{FA} is then inserted in Equation II.11b, to receive F_{RA}. The current load on the axles must not exceed the permissible ones:

$$F_{FA} \leq FA_{perm} \cdot g, \tag{II.13}$$

$$F_{RA} \leq RA_{perm} \cdot g, \tag{II.14}$$

$$F_{TA} \leq TA_{perm} \cdot g. \tag{II.15}$$

4.4 Consideration of Vehicle's and Trailer's Masses

The value of the permissible axle weights may be without the consideration of vehicle mass (m_{truck}). In this case, the mass must be respected in the formulas. The mass is added as additional force in Equation II.4b:

$$F_{FA} = \sum_{i=1|i\in R_v}^{n} \sum_{k=1}^{c_i} F_{i,k} + m_{truck} \cdot g - F_{RA}. \tag{II.4c}$$

Moreover, the mass creates a moment in the center of mass of the truck. The distance (r_{truck}) between the front axle and the center of mass of the truck must be given by the manufacturer. Thus, Equation II.5b needs to be updated as follows:

$$F_{RA} = \frac{1}{WB} \cdot (\sum_{i=1|i\in R_v}^{n} \sum_{k=1}^{c_i} M_{i,k} + m_{truck} \cdot g \cdot r_{truck}). \tag{II.5c}$$

In the case of semi-truck trailers, the masses of the tractor unit ($m_{tractor}$) and of the trailer ($m_{trailer}$) do not have to be included. In that case, the masses are added as forces in the Equations II.9b and II.11b, so that

$$F_{TA} = \sum_{i=1|i\in R_v}^{n} \sum_{k=1}^{c_i} F_{i,k} + m_{trailer} \cdot g - F_K. \tag{II.9c}$$

and

$$F_{RA} = F_K + m_{tractor} \cdot g - F_{FA}. \tag{II.11c}$$

The mass of the trailer and the tractor unit creates also a moment in their center of mass. The distance $r_{tractor}$ between the rear axle and the center of mass and the distance $r_{trailer}$ between the trailer axle and the center of mass must be also given by the manufacturer. The moments are added in the Equations II.10b and II.12b, resulting in

$$F_K = \frac{1}{l_{K|TA}} \cdot \left(\sum_{i=1|i\in R_v}^{n} \sum_{k=1}^{c_i} F_{i,k} \cdot r_{i,k} + m_{trailer} \cdot g \cdot r_{trailer} \right) \tag{II.10c}$$

and

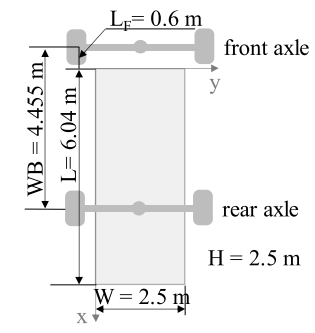
$$F_{FA} = \frac{1}{WB} \cdot (F_K \cdot l_{K|RA} + m_{tractor} \cdot g \cdot r_{tractor}). \tag{II.12c}$$

Other masses, such as the mass of the driver, the fuel, etc., may be added in the same way, provided that distances and masses are known.

4.5 Check Frequency

In the following, the check frequency for the Axle Weight Constraint is examined. According to Pollaris et al. (2016), by placing an item, the mass of the item cannot only act on an axle, but it can also relieve an axle. Therefore, when unloading an item, the load on the axles can exceed the permissible ones. Thus, it must be checked whether the constraint is fulfilled after each placement of an item. In the following, we will demonstrate this with a realistic example. The demand of four customers consists of one item each. We assume the truck is the model ML150E28FP of the manufacturer IVECO. The dimensions of the truck and of the items are shown in Fig. II.5. The load capacity D is 10,100 kg and the permissible load on the axles is FA_{perm} = 5,300 kg for the front axle and RA_{perm} = 10,700 kg for the rear axle.

Vehicle



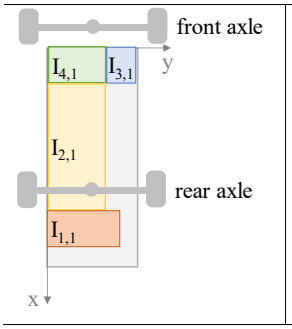
Items

Item	$m_{i,k}$ [kg]	$l_{i,k}$ [m]	$w_{i,k}$ [m]	$h_{i,k}$ [m]
$I_{1,1}$	2,000	1.00	2.00	1.60
$I_{2,1}$	1,000	3.50	1.60	1.60
$I_{3,1}$	3,000	1.00	0.80	1.60
$I_{4,1}$	4,100	1.00	1.60	1.60
	10,100			

Figure II.5: Vehicle's and Items' Dimensions

The customer visiting order for the route is $R_1 = \{1, 2, 3, 4\}$. Due to the height of the items, it is not possible to stack them. The load capacity D is complied with since the sum of the items' masses is equal to the load capacity D. Respecting the LIFO (C4) constraint, the items of the customers are unloaded in the reversed order ($order = \{I_{4,1}; I_{3,1}; I_{2,1}; I_{1,1}\}$).

To pack the items, the proposed DBLF algorithm is used (see Section 5.2). The items' positions, shown in Fig. II.6, would result if the Axle Weight Constraint would be checked once after loading all items of the route R_1.



Item	$m_{i,k}$ [kg]	$l_{i,k}$ [m]	$x_{i,k}$ [m]	$r_{i,k}$ [m]	$F_{i,k}$ [N]	$M_{i,k}$ [Nm]
$I_{4,1}$	4,100	1.00	0.00	1.10	40,221	44,243
$I_{3,1}$	3,000	1.00	0.00	1.10	29,430	32,373
$I_{2,1}$	1,000	3.50	1.00	3.35	9,810	32,864
$I_{1,1}$	2,000	1.00	4.50	5.60	19,620	109,872

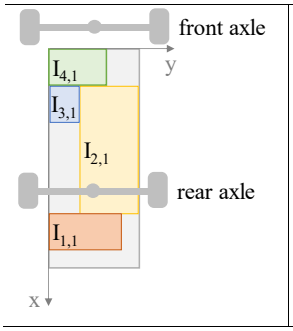
Figure II.6: Positions of items when checking the Axle Weight Constraint when all items have been loaded

The corresponding axle weights are shown in Table II.1. The load on the axles are below the permissible ones after loading all items into the vehicle's cargo space. Thus, the Axle Weight Constraint seems to be fulfilled. However, when unloading the item $I_{1,1}$, the front axle gets overloaded because the item $I_{1,1}$ relieves the front axle. The front axle is even overloaded after unloading item $I_{2,1}$. Consequently, the Axle Weight Constraint is not fulfilled for the current items' positions, shown in Fig. II.6.

Table II.1: Axle Weights for items' positions in Fig. II.6

Loaded items	Current Forces		Permissible Forces	
	F_{FA} [N]	F_{RA} [N]	$FA_{perm} \cdot g$ [N]	$RA_{perm} \cdot g$ [N]
$I_{4,1}$; $I_{3,1}$; $I_{2,1}$; $I_{1,1}$	49,844	49,237	51,993	104,967
$I_{4,1}$; $I_{3,1}$; $I_{2,1}$	**54,886**	24,575		
$I_{4,1}$; $I_{3,1}$	**52,453**	17,198		
$I_{4,1}$	30,290	9,931		

When checking the Axle Weight Constraint after each placement of an item, the items' positions shown in Fig. II.7 would result. The item $I_{4,1}$ would be still placed in the origin. The first position for item $I_{3,1}$ leads to an overload of the front axle as shown in Fig. II.6. Due to the check of the Axle Weight Constraint, this position would be rejected and the next possible position according to the DBLF approach would be tested, which is in front of item $I_{4,1}$. For this position, the front axle is not overloaded and the position is feasible.



Item	$m_{i,k}$ [kg]	$l_{i,k}$ [m]	$x_{i,k}$ [m]	$r_{i,k}$ [m]	$F_{i,k}$ [N]	$M_{i,k}$ [Nm]
$I_{4,1}$	4,100	1.00	0.00	1.10	40,221	44,243
$I_{3,1}$	3,000	1.00	1.00	2.10	29,430	61,803
$I_{2,1}$	1,000	3.50	1.00	3.35	9,810	32,864
$I_{1,1}$	2,000	1.00	4.50	5.60	19,620	109,872

Figure II.7: Positions of items when checking the Axle Weight Constraint after each item's placement

Table II.2 shows the calculated axle weights for the items' positions in Fig. II.7. Since the Axle Weight Constraint is fulfilled after each item has been unloaded, the axles are not overloaded at any point in the route.

To summarize, it is necessary to check the Axle Weight Constraint after each placement of an item, although the complexity of the algorithm increases.

Table II.2: Axle Weights for items' positions in Fig. II.7

Loaded items	**Current Forces**		**Permissible Forces**	
	F_{FA} [N]	F_{RA} [N]	$FA_{perm} \cdot g$ [N]	$RA_{perm} \cdot g$ [N]
$I_{4,1}$; $I_{3,1}$; $I_{2,1}$; $I_{1,1}$	43,238	55,843	51,993	104,967
$I_{4,1}$; $I_{3,1}$; $I_{2,1}$	48,280	31,181		
$I_{4,1}$; $I_{3,1}$	45,847	23,804		
$I_{4,1}$	30,290	9,931		

5 Hybrid Solution Approach

Since the 3L-CVRP can be interpreted as a combination of the Capacitated Vehicle Routing Problem (CVRP) and the 3D Container Loading Problem, we decompose the problem and use a separate algorithm for each subproblem. First, a set of routes is created which takes the routing constraints (R1, R2) into account. For each feasible route, the packing algorithm is then called, which tries to create a feasible packing plan considering the loading constraints. If no feasible packing plan can be created for a route, a new set of routes must be found. Both algorithms are described in detail in the following subsections, suitable line numbers are given in square brackets.

5.1 Routing Heuristic

The heuristic for solving the routing problem is based on the paper by Koch et al. (2018) modifying the Adaptive Large Neighborhood Search (ALNS) proposed by Ropke and Pisinger (2006b) and Ropke and Pisinger (2006a). The general framework and the modifications are described below. The algorithm is shown in Alg. II.1.

Algorithm II.1 Adaptive Large Neighbourhood Search

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

1: construct initial solution s_{init}
2: $s_{best} := s_{init}$
3: $s_{curr} := s_{init}$
4: **do**
5: select number of customers to be removed n_{rem}
6: select destroy operator *dest*
7: select repair operator *rep*
8: determine next solution $s_{next} := rep(dest(s_{curr}, n_{rem}))$
9: check acceptance of s_{next}
10: **if** s_{next} is accepted **then**
11: $s_{curr} := s_{next}$
12: **if** $f(s_{curr}) < f(s_{best})$ **then**
13: $s_{best} := s_{next}$
14: **end if**
15: **end if**
16: **if** it_p reached **then**
17: update selection probabilities for destroy and repair operators
18: **end if**
19: **while** one stopping criterion is not met

5.1.1 Initial Solution

The initial solution s_{init} is constructed [1] by means of the Savings Heuristic developed by Clarke and Wright (1964), where one route is created for each customer first. Then, the savings for merging routes are calculated. Starting with the highest savings, all feasible merges are carried out while respecting all constraints. Based on this initial solution, the ALNS determines other feasible improved solutions.

5.1.2 Stopping Criteria

Starting with the initial solution received by the Savings Heuristic, the ALNS tries to improve the current solution as long as not one of the following stopping criteria is met [19]:

- number of total iterations $iter_{max}$;
- number of iterations without improvement $iter_{impr}$;
- runtime limit t_{max}.

5.1.3 Iteration

A new solution s_{next} is generated by choosing randomly one destroy [6] and one repair operator [7] in each iteration. The destroy operator removes a number of customers n_{rem} from the current solution and the repair operator reinserts them [8]. The number of customers to be removed n_{rem} ($n_{min} \leq n_{rem} \leq n_{max}$) is determined randomly. The generated solution is checked with respect to meeting the routing constraints. The packing procedure (see Section 5.2) is called here for each route of the solution.

5.1.4 Destroy and Repair Operators

In the following, we give a short overview over the used destroy and repair operators. All operators are described in more detail in the references.

Shaw Destroy Operator The idea of this operator is to exclude related customers from a solution since it is easier to exchange related customers (see Shaw (1997)). The relatedness $rlt_{i,j}$ between a customer i and customer j is evaluated by means of the distance $co_{i,j}$, the demanded volume of the customers and, as shown by Demir et al. (2012), also the assignment to tours ($ass = 1$ if in same tour, -1 else). Each parameter is weighted by means of coefficients:

$$rlt_{i,j} = \omega_{co} \cdot co_{i,j} + \omega_{vol} \cdot |\sum_{k=1}^{c_i}(l_{i,k} \cdot w_{i,k} \cdot h_{i,k}) - \sum_{k=1}^{c_j}(l_{j,k} \cdot w_{j,k} \cdot h_{j,k})| + \omega_{st} \cdot st_{i,j} \tag{II.16}$$

The smaller the value of $rlt_{i,j}$, the more related the customers are. The first customer i to be removed is determined randomly, while in the next iterations, randomly a customer i is chosen out of a set containing all removed customers. Then the relatedness between the customer i and every not yet removed customer j is calculated and saved in a list, which is sorted in ascending order w.r.t. the relatedness. The most related customer is not necessarily selected from this list, as a determinism parameter dp ($dp \geq 1$) provides a degree of randomness (see Ropke and Pisinger (2006b)). The procedure is repeated until n_{rem} customers are excluded.

Random Destroy Operator As introduced in Ropke and Pisinger (2006b), the operator randomly removes customers out of the solution until n_{rem} customers have been removed in total.

Worst Destroy Operator In this approach proposed by Ropke and Pisinger (2006b), the cost of each customer is determined, which is the difference between the total travel distance of the current solution and the total travel distance of the solution if the customer would be completely removed. Then, the n_{rem} customers are removed which deteriorate the total travel distance of the solution the most.

Cluster Destroy Operator The first route is selected randomly. This route is then divided into two connected clusters (trees) by using a modified algorithm of Kruskal (1956) (see Koch et al. (2018)). One of the generated clusters is chosen randomly and its customers are removed. If n_{rem} is not reached, one already removed customer is selected. Based on this customer, its closest customer which is not included in an already removed route is searched. The procedure is repeated with this route until (at least) n_{rem} customers have been removed.

Neighbour Graph Destroy Operator As shown in Ropke and Pisinger (2006a), this operator is used to remove customers whose position in a tour seems to be “inappropriate". Thereby, the customers to be removed are selected by using historical information, which is stored in a complete, directed, weighted graph (neighbor graph). Each node represents one customer. The arc (i, j) is assigned the total travel distance of the best solution in which customer i is visited just before customer j. Initially, the arc weights are set sufficiently high. Whenever a new solution is found, the arc weights in the graph are updated if necessary.
For the selection of the customers to be removed, a score is calculated for each customer in the current solution. The score is determined by summing the arc weights in the neighbor graph corresponding to the neighbor configuration in the current solution. High scores indicate rather misplaced customers. As in the Shaw Destroy Operator, the parameter dp influences the process so that the customer with the highest score is not always selected.

Overlap Destroy Operator As proposed in Koch et al. (2018), for each tour, a bounding box is determined based on the minimum and maximum x- and y- coordinates of the customer locations. If two bounding boxes overlap, then the overlapping area is determined. Within the overlapping area, all intersecting edges between the tours are stored. Randomly, one pair of these intersecting edges is selected. All corresponding customers which also lay within the overlapping area are removed from the solution.

Inner Route Destroy Operator This operator is introduced in Koch et al. (2018) and checks if one tour is completely surrounded by another tour. Hereby, the bounding box for each tour is determined as described in the Overlap Destroy Operator. If one of these boxes lies completely within another one, then the inner tour is removed from the current solution. The outer tour is separated into two new tours, so that one tour contains the first half of the customers and the other tour contains the rest.

Intersection Destroy Operator This operator is proposed by Koch et al. (2018). For each tour, all intersecting edges are determined. Then, one intersecting pair is randomly selected and all four customers of this pair of edges are removed. This operator is repeated until n_{rem} customers are removed.

Tour Pair Destroy Operator As in the Overlap Destroy Operator, intersecting pairs of tours are first determined. Then, all customers of a randomly selected pair of tours are removed. The operator is introduced in Koch et al. (2018).

Greedy Repair Operator For each removed customer and each position in every tour, the insertion costs are determined (see Ropke and Pisinger (2006b). The customer is then inserted at the position in the tour causing the smallest increase in the total travel distance. Only feasible insertions obeying all constraints are considered.

Regret-2 and Regret-3 Repair Operators As proposed in Ropke and Pisinger (2006b), in each iteration and for each removed customer, a regret value is calculated, which is the change in the objective value when inserting the customer into the second best position of one tour and the best position of another tour (regret-2). In the case of regret-3, the regret value corresponds to the regret value received for regret-2 plus the difference between the total travel distance of the third best and the best insertion. Then, the customer with the largest regret value is selected and inserted. This is repeated until all customers have been tried for insertion.

Customers who cannot be inserted into a route due to the violation of a constraint are stored in a list and cause a penalization of the objective value as shown in Sec. 5.1.6. The selection probabilities for the destroy and repair operators are adjusted according to the improvement of the solution after a defined number of iterations it_p [16-18].

5.1.5 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 its weight 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:

$$score_{op} = iter_{best_{op}} \cdot \omega_{best} + iter_{impr_{op}} \cdot \omega_{impr} + iter_{e_{op}} \cdot \omega_e. \tag{II.17}$$

Hereby, ω_{best}, ω_{impr} and ω_e are coefficients. Then, the updated weighting wg_{op} can be calculated. A reaction factor r regulates the influence of the adaptions:

$$wg_{op} = wg_{op} \cdot (1 - r) + r \cdot \frac{score_{op}}{counter_{op}} \tag{II.18}$$

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

5.1.6 Objective Function

A solution s is evaluated with objective function f giving total routing costs in order to lead the further search. It considers the number of used vehicles v_{used} as well as the total travel distance $ttd(s)$:

$$f(s) = ttd(s) + pen_v \cdot \max(0, v_{used} - v_{max}) + pen_{miss} \cdot n_{miss}. \tag{II.19}$$

v_{max} is the number of maximal available vehicles and n_{miss} the number of customers that have not been dispatched yet. The purpose of the penalty term pen_v is to create a solution using at most v_{max} available vehicles. The purpose of the penalty term pen_{miss} is to lead the search to decrease the number of not dispatched customers (n_{miss}).

5.1.7 Solution Acceptance

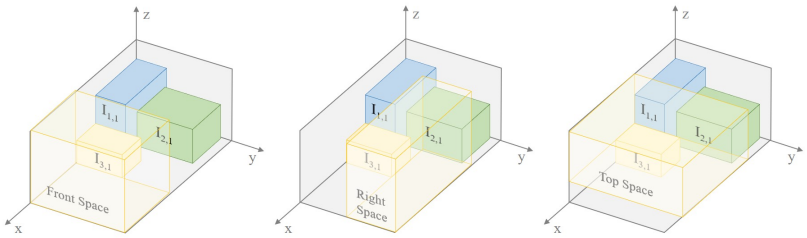
The smaller the objective function value, the better the solution. If the generated solution s_{next} is feasible according to all constraints and better than the current best-known s_{best} one, it is always accepted as current solution s_{curr}. A worse solution may be accepted depending on a Simulated Annealing Framework based on Kirkpatrick et al. (1983) [9]. The acceptance probability is adapted to the annealing process with a geometric cooling schedule. The best solution s_{best} is updated if its objective function value is higher than of the current solution s_{curr} [12-14].

5.2 Packing Heuristic

The packing heuristic is called by the routing heuristic for each route of a solution. It is based on the DBLF algorithm proposed by Karabulut and İnceoğlu (2005). 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 cargo space. The available free spaces in the cargo space are saved in a list. In the following, the point of origin of a Cartesian coordinate system is located in the deepest, bottom, leftmost point of the cargo space. The driver's cab is located behind it accordingly. The length, width and height of the cargo 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.
In the first step of the packing heuristic, the items of each customer are stored in the set IS in a sorted order observing 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).

The algorithm is shown in Alg. II.2. The order of the items in the packing sequence IS are reversed to the customer's visiting order [1].
Let S be a set containing unique cuboids representing the available free spaces in the cargo space after placing an item. Initially, this set consists of one space representing the total cargo space [2]. Consequently, the first item of the packing sequence IS 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 a selected space. Each space sp of the set is tested as possible item position until a feasible position is found obeying all loading constraints [7]. In comparison to Karabulut and İnceoğlu (2005), the set S contains not all available spaces inside the cargo space. Rather, three new spaces (Front, Right, Top), based on the feasible item placement, are created [9].



(a) Front Space (b) Right Space (c) Top Space

Figure II.8: New Spaces based on $I_{3,1}$

The *Front Space* is defined by the item's front edge (minimum x-value) and either the cargo space door or the nearest item in front of the item (maximum x-value). Then, the minimum and maximum values for the y-axis are determined by the cargo space or other items. After that, the minimum and maximum values for the z-axis are searched in the same way (see Fig. II.8a).
The *Right Space* is bounded along the y-axis by the left side of the item (minimum y-value) and either by the cargo space wall or by the rightmost item (maximum y-value). Based on these limitations, the minimum and maximum values for the x-axis are determined, then for the z-axis (see Fig. II.8b).
The *Top Space* is defined by the item's top surface and either the cargo space ceiling or an item overhanging over the current item I_p. In the next step, the minimum and maximum values for the y-axis and z-axis determined by the cargo space or other items (see Fig. II.8c). The three new spaces (Front, Right, Top) are included in the set S considering the DBL order.

If a feasible position is found for item I_p, all remaining free spaces are checked w.r.t an intersection with item I_p. If there are one or more intersections with item I_p, then the minimum and maximum values for the x-, y- and z-axis are decreased so that no intersection with item I_p occurs any more [14]. Due to this procedure, it is guaranteed that the item does not overlap with other items or with the cargo space walls (C1). Therefore, if an item can be placed within an available space according to the loading constraints C2-C8, an extra overlapping check for the Geometry constraint (C1) is not necessary. This is in contrast to the approach by Karabulut and İnceoğlu (2005), where an overlapping check between each item is performed.
The space sp, in which the item I_p is placed, is removed from the set [11]. Only spaces which are large and high enough for the smallest dimensions of any item among all items of the route IS, 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 IS 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 [18].

Algorithm II.2 Deepest-Bottom-Left-Fill with Spaces

Input: Instance Data
Output: Feasible placements for items

```
initialize sorted sequence of items IS
initialize set of unique available spaces S
for each item I_p ∈ IS do
    for each space sp ∈ S do
        for each permitted orientation do
            if item I_p fits in space sp then
                if placement is feasible w.r.t. loading constraints then
                    save placement for I_p
                    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 w.r.t. l_min and h_min 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
```

If all spaces are checked and no feasible position for item I_p was found [24], the route is not feasible and is rejected. Thus, a new set of routes must be generated by the ALNS.

6 Computational Experiments

In this section, the impact of the Axle Weight Constraint on 2L-CVRP and 3L-CVRP instances is evaluated as well as the efficiency of the hybrid algorithm. This is done by testing instance sets from the literature with and without the Axle Weight Constraint (C8) and comparing the results with the benchmarks. The hybrid heuristic 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 ALNS for solving the routing problem and the loading constraints are parameterized as shown in Table II.3. The parameters for the routing heuristic are adopted from Koch et al. (2018).

Table II.3: Routing and Loading Parameters

Parameter	Usage	Description	Value
$iter_{max}$	ALNS	Maximal number of iterations	25,000
$iter_{impr}$	ALNS	Maximal number of iterations without improvement	8,000
$iter_p$	ALNS	Number of iterations for updating probabilities for destroy and repair operators	100
ω_{co}	ALNS	Coefficient for Shaw Destroy	6
ω_{vol}	ALNS	Coefficient for Shaw Destroy	3
ω_{st}	ALNS	Coefficient for Shaw Destroy	6
dp	ALNS	Determinism Parameter	6
ω_{best}	ALNS	Coefficient for determination of the operator score	50
ω_{impr}	ALNS	Coefficient for determination of the operator score	10
ω_e	ALNS	Coefficient for determination of the operator score	5
r	ALNS	Reaction factor	0.8
t_{max}	ALNS	Time limit [min]	60
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	$0.4n$
co_{max}	Objective Function	Maximal distance between two customers in instance	$max_{i,j \in N} co_{i,j}$
pen_v	Objective Function	Penalty term for each surplus vehicle	$10 \cdot co_{max}$
pen_{miss}	Objective Function	Penalty term for missing customers	$10 \cdot co_{max}$
α	Vertical Stability	Minimal support ratio	0.75

6.2 Instances

We have tested our approach on four instance sets (see Table II.4). The first instance set comes from Pollaris et al. (2016) and consists of 128 2L-CVRP instances, varying the number of customers in the network (10, 15, 20 and 25). The second instance set was created by Pollaris et al. (2017) and also deals with the 2L-CVRP. This set contains 96 instances with 50, 75 or 100 customers. The third instance set concerns the 3L-CVRP and was proposed by Gendreau et al. (2006). The number of customers is ranging between 15 and 100. Moreover, we have created a new instance set considering Axle Weights of Semi-Truck Trailers, which can be found here: https://doi.org/10.24352/UB.OVGU-2021-023. All instances are tested five times.

Table II.4: Overview of tested instances

authors	number of instances	problem	number of customers
Pollaris et al. (2016)	128	2L-CVRP	[10, 15, 20, 25]
Pollaris et al. 2017	96	2L-CVRP	[50, 75, 100]
Gendreau et al. (2006)	27	3L-CVRP	[15-100]
Our Instance set	80	3L-CVRP	[30, 60, 90, 120]

For both 2L-CVRP instance sets, four different problem classes are created for each network by varying the number of items per customer (c_i) and the mass of demanded items (mc_i) (see Table II.5). There is a low ($4 \leq c_i \leq 7$) and a high ($1 \leq c_i \leq 15$) variation in the number of items per customer i. The mass of demanded items per customer i is categorized in heavy pallets ($1,000 \leq \frac{mc_i}{c_i} \leq 1,500$) and a mix of heavy and light pallets ($100 \leq \frac{mc_i}{c_i} \leq 500$).

Table II.5: Problem classes for Pollaris et al. (2016, 2017) Instances

		composition of demand: heavy pallets	composition of demand: heavy and light pallets
demand variation:	low	class 1	class 3
	high	class 2	class 4

As shown in Section 3, the 2L-CVRP considers the loading constraints Geometry (C1), Orthogonality (C2), Load Capacity (C3), LIFO (C4) and the Axle Weight Constraint (C8), whereas in the 3L-CVRP, the Rotation (C5), the Minimal Supporting Area (C6) and the Fragility (C7) are additionally taken into account.
There were no values for the axle weights assigned to the vehicles in the instances by Gendreau et al. (2006). Thus, the two-axle truck ML180E by the manufacturer IVECO was chosen. The proportion factor was calculated from the truck cargo load and the load capacity D of the instance. Then, the axle weights were proportionally scaled in order to receive a realistic proportion between vehicle load capacity and axle weights.
The newly created instance set deals with Axle Weights for Semi-Truck Trailers. The number of customers is either 30, 60, 90 or 120, the number of items either 200 or 400 and the number of item types either 10 or 100. The Semi-Truck Trailer values take into account the directives EU 96/53/EC and EU 1230/2012. Under maximum load and maximum utilisation of the load volume, the average density is 0.25 $\frac{kg}{dm^3}$. Therefore, the densities of the item types vary between 0.1 and 0.5 $\frac{kg}{dm^3}$.

6.3 Results

In the following, we compare the results of our hybrid heuristic algorithm with instances from the literature and our instance set. Moreover, we show the impact of the Axle Weight Constraint on the total travel distance. Note again that the objective is to minimize the total travel distance. All results are published online via http://www.github.com/CorinnaKrebs/Results. In addition, we provide a solution validator to check the feasibility of packing plans concerning the Axle Weight Constraint. The solution validator is available in Java and C++ via http://www.github.com/CorinnaKrebs/SolutionValidator.

6.3.1 Results for the Pollaris et al. (2016) Instances

In Pollaris et al. (2016), the 2L-CVRP with the Axle Weight Constraint is investigated so that no stacking or vertical stability constraint is required. The pallets must be alternately packed in two rows without rotation. We have adapted our packing algorithm accordingly. Another additional constraint is that the driving axle carries at least 25% of the current load. Pollaris et al. (2016) use CPLEX for solving the problem with a time limit of 2 hours, within which not all instances could be solved. For the instances without a solution, our corresponding results are excluded from the difference calculation to enable fair comparison. The detailed results are in the Appendix. Table II.6 shows the summarized best results for each network and each class.

Table II.6: Summarized Best Results for Pollaris et al. (2016) Instance Set

		without Axle Weight Constraint						**with Axle Weight Constraint**					
		Pollaris et al. (2016)		**ALNSxDBLF**				Pollaris et al. (2016)		**ALNSxDBLF**			
n	**Class**	*ttd*	*time* [s]	*ttd*	**diff.** *ttd*	*time* [s]	**diff.** *time*	*ttd*	*time* [s]	*ttd*	**diff.** *ttd*	*time* [s]	**diff.** *time*
10													
	1	342.40	2.75	342.42	0.01%	0.18	-93.50%	354.30	52.13	360.41	1.72%	0.18	-99.65%
	2	397.00	7.56	396.98	0.00%	0.19	-97.45%	418.90	26.13	423.37	1.07%	0.49	-98.11%
	3	330.20	2.44	330.16	-0.01%	0.16	-93.38%	333.10	7.06	335.64	0.76%	0.18	-97.45%
	4	407.00	2.19	406.90	-0.02%	0.19	-91.31%	413.70	9.25	420.11	1.55%	0.57	-93.80%
15													
	1	355.60	75.50	355.70	0.03%	0.35	-99.53%	366.30	1,514.50	373.38	1.93%	0.34	-99.98%
	2	372.20	391.33	372.18	-0.01%	0.47	-99.88%	394.80	467.17	394.36	-0.11%	0.95	-99.80%
	3	437.80	76.75	437.97	0.04%	0.30	-99.61%	441.00	255.63	438.29	-0.61%	0.32	-99.87%
	4	594.60	63.00	594.58	0.00%	0.48	-99.24%	610.40	73.00	625.58	2.49%	1.02	-98.60%
20													
	1	215.80	121.67	215.80	0.00%	0.58	-99.52%	216.00	682.33	220.37	2.02%	0.61	-99.91%
	2	474.10	633.00	474.07	-0.01%	0.99	-99.84%	485.40	2,505.60	545.69	12.42%	2.44	-99.90%
	3	487.90	1,388.14	487.91	0.00%	0.67	-99.95%	488.00	3,088.71	489.58	0.32%	0.67	-99.98%
	4	581.00	268.00	581.01	0.00%	0.85	-99.68%	593.50	947.33	617.38	4.02%	1.94	-99.80%
	Total	**4,995.60**	**233.25**	**4,995.67**	**0.00%**	**0.41**	**-99.82%**	**5,115.40**	**705.62**	**5,244.14**	**2.52%**	**0.74**	**-99.89%**

The total travel distance is given in column *ttd*. The runtime in seconds is displayed in column *time* [s]. The relative differences between the benchmark results and the results received by our hybrid algorithm are calculated in diff. *ttd*. The column diff. *time* shows the difference for the average runtime.
Regarding the results without the Axle Weight Constraint, our hybrid algorithm receives the

same results as the benchmark. The small deviations are due to rounding inaccuracies and the necessary classical multidimensional scaling, where the distance matrix is recalculated to customer coordinates. The disregard of the axle weights leads to overloaded trucks in nearly every instance. Therefore, the total travel distance increases compared to the results without the Axle Weight Constraint. Compared to the benchmark, our hybrid algorithm does not achieve the same results when considering the Axle Weight Constraint. On average, the total travel distance increases by 2.52% compared to the benchmark. In Pollaris et al. (2016), the results found without the Axle Weight Constraint are checked whether fulfilling the Axle Weight Constraint (lazy constraint). The best results are then presented as results with Axle Weight Constraint. Therefore, the results with the Axle Weight Constraint are not necessarily the optimal results. For some instances, our hybrid algorithm finds better results. Furthermore, our hybrid heuristic algorithm achieves all results with a runtime saving of 99.8% on average.

The following Table II.7 illustrates a comparison of our results in order to evaluate the demand variation and the demand weight.

Table II.7: Comparison of our results for Pollaris et al. (2016) instances

		Without Axle Weight Constraint		**With Axle Weight Constraint**			
		best	**avg.**	**best**		**avg.**	
n	Class	*ttd*	*ttd*	*ttd*	diff.	*ttd*	diff.
10							
	1	342.42	342.43	360.41	5.25%	360.41	5.25%
	2	396.98	396.99	423.37	6.65%	423.37	6.65%
	3	330.16	330.14	335.64	1.66%	335.83	1.72%
	4	406.90	406.90	420.11	3.25%	420.10	3.24%
15							
	1	355.70	355.71	373.38	4.97%	373.38	4.97%
	2	372.18	372.18	394.36	5.96%	394.36	5.96%
	3	437.97	437.96	438.29	0.07%	438.29	0.08%
	4	594.58	594.58	625.58	5.21%	625.59	5.22%
20							
	1	215.80	215.80	220.37	2.12%	220.53	2.19%
	2	474.07	474.06	545.69	15.11%	546.79	15.34%
	3	487.91	487.91	489.58	0.34%	489.58	0.34%
	4	581.01	581.01	617.38	6.26%	617.38	6.26%
Total		**4,995.67**	**4,995.67**	**5,244.14**	**4.97%**	**5,245.61**	**5.00%**

Generally, there is no deviation between the best and the average of five runs for the results without considering the axle weights. Consequently, the hybrid algorithm always finds the best solution. If the axle weights are taken into account, the difference between the best and the average result is only 0.3% points. On average, considering the Axle Weight Constraint leads to an increase of the total travel distance of around 5%.
The instances of class 2 are the most difficult to solve since the customers demand a large number of heavy pallets. Especially, for instances with a number of 20 customers, the total distance increases by 15.34% on average. The easiest class to solve is class 3. For these instances, the Axle Weight Constraint influences the results in a minor way so that the total travel distance increases by less than one percent on average.
The impact of the variation of the number of items on the objective value can be evaluated by comparing the results of classes 1 with 2 and of classes 3 with 4. The impact of heavy pallets on the objective value can be examined by comparing the classes 1 with 3 and classes 2 with 4. As the results show, the mass of items influences the Axle Weight Constraint more than the demand variation (on average several percent points) and has therefore a higher impact on the total travel distance.
For instances with 25 customers, the approach of Pollaris et al. (2016) finds no solutions within the 2 hours time limit. Moreover, two instances were not tested: In instance with 20 customers, class 1, no. 2, the demanded mass per customer is not given, and in instance with 15 customers, class 2, no 2, the distance matrix is not consistent.

6.3.2 Results for the Pollaris et al. (2017) Instances

In Pollaris et al. (2017), the 2L-CVRP is solved by means of an Iterated Local Search approach with Sequence-Based Pallet Loading. It also applies to this instance set that the pallets are packed alternately and the driving axle must carry 25% of the current load. In Table II.8, the best results received by Pollaris et al. (2017) including the Axle Weight Constraints are compared with the results obtained by our hybrid heuristic algorithm. The detailed results are in the Appendix.

Table II.8: Summarized Best Results for Pollaris et al. (2017) Instance Set

		without Axle Weight Constraint				**with Axle Weight Constraint**				
		Pollaris et al. (2017)	**ALNSxDBLF**			**Pollaris et al. (2017)**		**ALNSxDBLF**		
n	**Class**	*ttd*	*ttd*	diff. *ttd*	*time* [s]	*ttd*	*time* [s]	*ttd*	diff. *ttd*	*time* [s]
50										
	1	1,186.30	1,185.91	-0.03%	9.72	1,215.40	68.38	1,240.47	2.06%	13.57
	2	1,478.20	1,478.78	0.04%	5.86	1,513.00	55.88	1,790.09	18.31%	15.98
	3	1,171.80	1,172.38	0.05%	8.61	1,176.00	97.13	1,177.68	0.14%	9.92
	4	1,614.60	1,614.25	-0.02%	6.09	1,649.30	55.38	1,768.51	7.23%	12.15
75										
	1	1,645.90	1,643.97	-0.12%	32.92	1,684.70	213.13	1,705.02	1.21%	48.10
	2	2,335.90	2,332.95	-0.13%	22.89	2,387.50	303.00	2,728.22	14.27%	38.45
	3	1,665.30	1,664.86	-0.03%	33.63	1,677.90	212.75	1,666.72	-0.67%	38.16
	4	2,426.80	2,426.20	-0.02%	20.07	2,457.40	309.88	2,592.33	5.49%	31.25
100										
	1	2,158.90	2,158.35	-0.03%	70.54	2,237.80	285.13	2,272.93	1.57%	103.28
	2	2,945.50	2,943.44	-0.07%	48.40	3,002.20	379.63	3,545.08	18.08%	98.07
	3	2,106.10	2,095.47	-0.50%	74.02	2,137.40	270.63	2,103.57	-1.58%	80.35
	4	3,068.20	3,066.23	-0.06%	50.23	3,143.60	399.38	3,430.63	9.13%	67.09
	Total	**23,803.50**	**23,782.79**	**-0.09%**	**31.91**	**24,282.20**	**220.85**	**26,021.23**	**7.16%**	**46.36**

Again, our hybrid algorithm achieves the same or even slightly better results if the Axle Weight Constraint is not included. When considering the constraint, the total travel distance increases by 7.16% compared to the benchmark. This is mainly due to instances of class 2, where the total travel distance increases by up to 18.31% compared to the benchmark. The results of Pollaris et al. (2017) were achieved with an average runtime of 220.65 seconds. Our algorithm only needed 46.46 seconds for the results (-78.95%).
Concerning the impact of the classes on the total travel distance when including the Axle Weight Constraint, findings similar to those above made for Pollaris et al. (2016) can be drawn.

6.3.3 Results for the Gendreau et al. (2006) Instances

In Table II.9, the results for the tested 3L-CVRP instances developed by Gendreau et al. (2006) are shown. Since this paper considers the Axle Weight Constraint for the first time in the 3L-CVRP, the comparison with the benchmark is based on the results without including this constraint.
For the comparison of the results without the Axle Weight Constraint, we use the best known results as listed in Escobar-Falcón et al. (2015). Our hybrid heuristic algorithm finds in 3 of 27 instances the best results. On average, the total travel distance increases by 5.91%.

Our analysis shows that without considering the axle weights, at least one of the axles would be overloaded for 25% instances. When considering the Axle Weight Constraint, the total travel distance rises only by 2.09% on average. In 2 out of 27 instances, the hybrid algorithm finds even better results when considering the Axle Weight Constraint, which is possibly due to the random parameters in the ALNS (e.g. random selection of operators). Moreover, the runtime decreases by 17.06% on average. The reason is that the load on the axles can be calculated relatively quickly, so that infeasible positions for items are detected faster and the ALNS can terminate earlier.

6.3.4 Results for Semi-Truck Trailer Instances

As this new instance set varies systematically in the number of customers, item types and items, the following Table II.10 presents the results for each group.
The difference between the best results and the average results is generally rather small (around 1%). When comparing the results with and without the Axle Weight Constraint, then the total travel distance increases by 5.31% on average. If the Axle Weight Constraint is not included, the runtime is 3600 seconds for all results. In contrast, when including the constraint, the runtime is

Table II.9: Results for Gendreau et al. (2006) Instances

	without Axle Weight C.				with Axle Weight C.		Deviation due to Axle Weights	
	Best Known	ALNSxDBLF			ALNSxDBLF			
no.	*ttd*	*ttd*	diff. *ttd*	*time*	*ttd*	*time*	*ttd*	*time*
1	302.02	302.02	0.00%	3.39	327.15	1.76	8.32%	-48.08%
2	334.96	334.96	0.00%	0.32	336.52	0.44	0.47%	37.50%
3	381.37	410.39	7.61%	24.03	403.07	5.36	-1.78%	-77.69%
4	437.19	440.68	0.80%	1.66	465.25	0.74	5.58%	-55.42%
5	436.48	454.14	4.05%	9.95	465.79	7.84	2.57%	-21.21%
6	498.16	498.32	0.03%	3.15	512.64	1.28	2.87%	-59.37%
7	767.46	788.46	2.74%	15.78	800.29	10.98	1.50%	-30.42%
8	804.75	845.51	5.06%	24.86	848.01	21.57	0.30%	-13.23%
9	630.13	658.43	4.49%	3.15	706.33	3.38	7.27%	7.30%
10	820.35	843.08	2.77%	118.34	865.73	83.84	2.69%	-29.15%
11	772.85	800.99	3.64%	94.66	829.77	82.20	3.59%	-13.16%
12	614.59	630.72	2.62%	10.19	653.05	6.25	3.54%	-38.67%
13	2,608.70	2,735.06	4.84%	128.38	2,767.62	93.86	1.19%	-26.89%
14	1,368.40	1,464.80	7.04%	138.41	1,514.53	101.57	3.40%	-26.62%
15	1,341.10	1,377.22	2.69%	150.67	1,429.03	100.04	3.76%	-33.60%
16	698.61	708.65	1.44%	4.91	719.24	6.16	1.49%	25.46%
17	866.40	866.40	0.00%	10.63	897.89	10.00	3.63%	-5.93%
18	1,207.70	1,283.87	6.31%	538.38	1,333.53	638.19	3.87%	18.54%
19	741.74	823.38	11.01%	612.78	838.16	470.68	1.80%	-23.19%
20	587.95	622.23	5.83%	2,200.53	631.15	1,894.11	1.43%	-13.92%
21	1,086.20	1,168.13	7.54%	2,970.01	1,177.80	2,173.63	0.83%	-26.81%
22	1,147.80	1,267.76	10.45%	2,515.36	1,295.21	1,491.06	2.17%	-40.72%
23	1,127.90	1,214.88	7.71%	2,849.25	1,235.07	1,673.42	1.66%	-41.27%
24	1,114.10	1,231.14	10.51%	1,546.01	1,265.30	870.15	2.77%	-43.72%
25	1,407.40	1,524.14	8.29%	3,600.00	1,553.06	3,600.00	1.90%	0.00%
26	1,585.50	1,720.73	8.53%	3,600.00	1,721.82	3,600.00	0.06%	0.00%
27	1,529.90	1,695.06	10.80%	3,600.00	1,676.22	3,600.00	-1.11%	0.00%
Total	**934.06**	**989.30**	**5.91%**	**917.59**	**1,009.97**	**761.06**	**2.09%**	**-17.06%**

Table II.10: Our Results for Semi-Truck Trailer Instances

	without Axle Weight Constraint			with Axle Weight Constraint							
	best	avg.		best				avg.			
n	*ttd*	*ttd*	*time* [s]	*ttd*	**diff.** *ttd*	*time* [s]	**diff.** *time*	*ttd*	**diff.** *ttd*	*time* [s]	**diff.** *time*
30	10,568.04	10,663.91	3,600.00	11,739.73	11.09%	417.89	-88.39%	11,743.87	10.13%	452.15	-87.44%
60	13,552.04	13,698.01	3,600.00	14,677.82	8.31%	3,005.83	-16.50%	14,701.15	7.32%	3,018.14	-16.16%
90	15,088.21	15,306.96	3,600.00	15,941.51	5.66%	3,541.41	-1.63%	16,039.74	4.79%	3,554.99	-1.25%
120	16,394.39	16,766.47	3,600.00	16,791.14	2.42%	3,600.00	0.00%	16,946.13	1.07%	3,600.00	0.00%
m											
200	23,719.11	24,078.26	3,600.00	24,536.00	3.44%	2,464.86	-31.53%	24,621.66	2.26%	2,487.75	-30.90%
400	31,883.57	32,357.09	3,600.00	34,614.20	8.56%	2,817.70	-21.73%	34,809.23	7.58%	2,824.89	-21.53%
types											
10	27,334.14	27,738.60	3,600.00	29,441.28	7.71%	2,579.94	-28.34%	29,566.67	6.59%	2,584.66	-28.20%
100	28,268.54	28,696.75	3,600.00	29,708.91	5.10%	2,702.62	-24.93%	29,864.22	4.07%	2,727.98	-24.22%
Total	**55,602.68**	**56,435.35**	**3,600.00**	**59,150.20**	**6.38%**	**2,641.28**	**-26.63%**	**59,430.89**	**5.31%**	**2,656.32**	**-26.21%**

around 2650 seconds on average and therefore reduced by 26.21%. As shown before, due to the limitation of the solution space, the ALNS terminates earlier than without the constraint.
Furthermore, the impact of the Axle Weight Constraint on the total travel distance tends to decrease by around 3% points per 30 customers as the number of customers increases (see avg. or best results with Axle Weight Constraint, column diff. *ttd*). The reason is that with a lower number of customers, the number of demanded items per customer is higher. Therefore, it is more difficult to pack all demanded items of one customer into one vehicle. Also the runtime increases along with the number of customers. Remarkable is the runtime reduction due to the Axle Weight Constraint for instances with 30 customers, which is 87.44% on average.
Moreover, with increasing number of items, the impact of the Axle Weight Constraint on the total travel distance increases noticeably by around 5% points and the runtime by 10% points. Interestingly, the results show that rather homogeneous items (less item types) have a slight negative impact on the total travel distance (around 2.5%) and the runtime (around 4%). We assume that the total used volume of the cargo space is higher in the case of homogeneous items so that a higher load is acting on the axles.

To summarize the presented results, the Axle Weight Constraint leads in general to only a slight increase of the total travel distance. In some cases, even better results can be achieved and the constraint can lead to a significant reduction of the runtime. Since the results without the consideration of the Axle Weight Constraint can lead to overloaded axles, the constraint should be an elementary part of container loading problems.

7 Conclusion

In this paper, we introduced two flexible approaches based on the Science of Statics for the consideration of axle weights of trucks with and without trailers. We showed with an example the necessity of checking the Axle Weight Constraint after each item placement. Computational experiments based on instances from the literature show that without the consideration of the Axle Weight Constraint, an overload of at least one axle would mostly result. The examination of the 2L-CVRP instances showed that the total travel distance increases significantly when the customers demand heavy pallets. When taking the Axle Weight Constraint into account, the deterioration of the total travel distance is small (on average: between 2% and 7%), depending on the instance set. In the case of 3L-CVRP instance sets, there are even significant positive effects on the average runtime leading to a decrease of up to 26%. With regard to the negative consequences of an overloaded axle (increased road surface erosion and extended braking distance), the mostly small decline of the objective value, the shorter runtime and the easy implementation, we recommend the consideration of axle weights in future approaches for the Container Loading Problems. For future work, we plan to integrate the Axle Weight Constraint directly in the packing algorithm to improve the selection process for the items' positions. In addition, items with high densities will be considered.

Conflict of interest

The authors declare that they have no conflict of interest.

References

Alonso, M., Alvarez-Valdes, R., Iori, M., and Parreño, F. (2019). "Mathematical models for Multi Container Loading Problems with practical constraints". In: *Computers and Industrial Engineering* vol. 127, pp. 722–733. ISSN: 0360-8352. DOI: https://doi.org/10.1016/j.cie.2018.11.012. URL: http://www.sciencedirect.com/science/article/pii/S0360835218305527.

Alonso, M., Alvarez-Valdes, R., Iori, M., Parreño, F., and Tamarit, J. (2017). "Mathematical models for multicontainer loading problems". In: *Omega* vol. 66, pp. 106–117. ISSN: 0305-0483. DOI: https://doi.org/10.1016/j.omega.2016.02.002. URL: http://www.sciencedirect.com/science/article/pii/S0305048316000335.

Blower, D. and Woodrooffe, J. (2012). "Survey of the status of truck safety: Brazil, China, Australia, and the United States". In.

Bortfeldt, A. (2012). "A hybrid algorithm for the capacitated vehicle routing problem with three-dimensional loading constraints". In: *Computers and Operations Research* vol. 39, no. 9, pp. 2248–2257. ISSN: 03050548. DOI: 10.1016/j.cor.2011.11.008. URL: http://dx.doi.org/10.1016/j.cor.2011.11.008.

Clarke, G. and Wright, J. W. (1964). "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points". In: *Operations Research* vol. 12, no. 4, pp. 568–581. ISSN: 0030364X, 15265463. DOI: 10.1287/opre.12.4.568. URL: http://www.jstor.org/stable/167703.

Demir, E., Bektaş, T., and Laporte, G. (2012). "An adaptive large neighborhood search heuristic for the Pollution-Routing Problem". In: *European Journal of Operational Research* vol. 223, no. 2, pp. 346–359. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2012.06.044. URL: https://www.sciencedirect.com/science/article/pii/S0377221712004997.

Escobar-Falcón, L. M., Álvarez-Martínez, D., Granada-Echeverri, M., Escobar-Velásquez, J. W., and Romero-Lázaro, R. A. (2015). "A matheuristic algorithm for the three-dimensional loading capacitated vehicle routing problem (3L-CVRP)". In: *Revista Facultad De Ingenieria-universidad De Antioquia*, pp. 9–20.

Fuellerer, G., Doerner, K. F., Hartl, R. F., and Iori, M. (2010). "Metaheuristics for vehicle routing problems with three-dimensional loading constraints". In: *European Journal of Operational Research* vol. 201, no. 3, pp. 751–759. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2009.03.046. URL: http://www.sciencedirect.com/science/article/pii/S0377221709002252.

Gendreau, M., Iori, M., Laporte, G., and Martello, S. (2006). "A Tabu Search Algorithm for a Routing and Container Loading Problem". In: *Transportation Science* vol. 40, no. 3, pp. 342–350. ISSN: 0041-1655. DOI: 10.1287/trsc.1050.0145. URL: http://pubsmisc.informs.org/doi/abs/10.1287/trsc.1050.0145.

Iori, M., Salazar González, J. J., and Vigo, D. (May 2007). "An Exact Approach for the Vehicle Routing Problem with Two-Dimensional Loading Constraints". In: *Transportation Science* vol. 41, pp. 253–264. DOI: 10.1287/trsc.1060.0165.

Karabulut, K. and İnceoğlu, M. M. (2005). "A Hybrid Genetic Algorithm for Packing in 3D with Deepest Bottom Left with Fill Method". In: *Advances in Information Systems*. Ed. by Yakhno, T. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 441–450. ISBN: 978-3-540-30198-1.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). "Optimization by simulated annealing." In: *Science* vol. 220 4598, pp. 671–80. DOI: 10.1126/science.220.4598.671.

Koch, H., Bortfeldt, A., and Wäscher, G. (Feb. 2018). "A hybrid algorithm for the vehicle routing problem with backhauls, time windows and three-dimensional loading constraints". In: *OR Spectrum* vol. 40. DOI: 10.1007/s00291-018-0506-6.

Kruskal, J. B. (1956). "On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem". In: *Proceedings of the American Mathematical Society* vol. 7, no. 1, pp. 48–50. ISSN: 00029939, 10886826. URL: http://www.jstor.org/stable/2033241.

Lim, A., Ma, H., Qiu, C., and Zhu, W. (2013). "The single container loading problem with axle weight constraints". In: *International Journal of Production Economics* vol. 144, no. 1, pp. 358–369. ISSN: 0925-5273. DOI: https://doi.org/10.1016/j.ijpe.2013.03.001. URL: http://www.sciencedirect.com/science/article/pii/S0925527313001084.

Pais, J., I. R. Amorim, S., and Minhoto, M. (Sept. 2013). "Impact of Traffic Overload on Road Pavement Performance". In: *Journal of Transportation Engineering* vol. 139, pp. 873–879. DOI: 10.1061/(ASCE)TE.1943-5436.0000571.

Pollaris, H., Braekers, K., Caris, A., Janssens, G., and Limbourg, S. (Mar. 2017). "Iterated local search for the capacitated vehicle routing problem with sequence-based pallet loading and axle weight constraints". In: *Networks* vol. 69, pp. 304–316. DOI: 10.1002/net.21738.

Pollaris, H., Braekers, K., Caris, A., Janssens, G. K., and Limbourg, S. (June 2016). "Capacitated vehicle routing problem with sequence-based pallet loading and axle weight constraints". In: *EURO Journal on Transportation and Logistics* vol. 5, no. 2, pp. 231–255. ISSN: 2192-4384. DOI: 10.1007/s13676-014-0064-2. URL: https://doi.org/10.1007/s13676-014-0064-2.

Ropke, S. and Pisinger, D. (2006a). "A unified heuristic for a large class of Vehicle Routing Problems with Backhauls". In: *European Journal of Operational Research* vol. 171, no. 3. Feature Cluster: Heuristic and Stochastic Methods in Optimization Feature Cluster: New Opportunities for Operations Research, pp. 750–775. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2004.09.004. URL: http://www.sciencedirect.com/science/article/pii/S0377221704005831.

— (2006b). "An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows". In: *Transportation Science* vol. 40, no. 4, pp. 455–472. DOI: 10.1287/trsc.1050.0135. URL: https://doi.org/10.1287/trsc.1050.0135.

Shaw, P. (1997). *A New Local Search Algorithm Providing High Quality Solutions to Vehicle Routing Problems*.

Tarantilis, C. D., Zachariadis, E. E., and Kiranoudis, C. T. (June 2009). "A Hybrid Metaheuristic Algorithm for the Integrated Vehicle Routing and Three-Dimensional Container-Loading Problem". In: *IEEE Transactions on Intelligent Transportation Systems* vol. 10, no. 2, pp. 255–271. ISSN: 1524-9050. DOI: 10.1109/TITS.2009.2020187.

Wei, L., Zhang, Z., and Lim, A. (Nov. 2014). "An Adaptive Variable Neighborhood Search for a Heterogeneous Fleet Vehicle Routing Problem with Three-Dimensional Loading Constraints".

In: *IEEE Computational Intelligence Magazine* vol. 9, no. 4, pp. 18–30. ISSN: 1556-603X. DOI: 10.1109/MCI.2014.2350933.

Appendix

Table II.11: Pollaris et al. (2016) Instances with 10 customers

		without Axle Weight Constraint								with out Axle Weight Constraint							
		best					avg.			best					avg.		
		Pollaris et al. (2016)		ALNSxDBLF			ALNSxDBLF			Pollaris et al. (2016)		ALNSxDBLF			ALNSxDBLF		
Class	No	*ttd*	*time* [s]	*ttd*	diff. *ttd*	*time* [s]	*ttd*	*time* [s]		*ttd*	*time* [s]	*ttd*	diff. *ttd*	*time* [s]	*ttd*	*time* [s]	
1																	
	1	38.40	1.00	38.43	0.08%	0.19	38.43	0.20		45.20	250.00	45.17	-0.07%	0.26	45.17	0.37	
	2	38.50	1.00	38.49	-0.03%	0.14	38.49	0.16		38.50	3.00	38.49	-0.03%	0.15	38.49	0.17	
	3	39.30	6.00	39.35	0.13%	0.20	39.35	0.22		39.50	8.00	40.19	1.74%	0.18	40.19	0.24	
	4	41.90	2.00	41.85	-0.12%	0.23	41.85	0.31		45.70	97.00	48.49	6.11%	0.19	48.49	0.27	
	5	51.70	4.00	51.71	0.01%	0.16	51.71	0.18		51.70	6.00	53.72	3.90%	0.18	53.72	0.20	
	6	43.40	4.00	43.42	0.05%	0.22	43.42	0.26		44.20	36.00	44.24	0.09%	0.18	44.24	0.24	
	7	45.20	3.00	45.22	0.05%	0.15	45.22	0.20		45.20	8.00	45.86	1.47%	0.16	45.86	0.19	
	8	44.00	1.00	43.96	-0.10%	0.14	43.96	0.22		44.30	9.00	44.25	-0.11%	0.17	44.25	0.22	
2																	
	1	41.20	7.00	41.19	-0.01%	0.15	41.19	0.23		44.30	77.00	44.34	0.09%	0.19	44.34	0.28	
	2	44.70	1.00	44.72	0.04%	0.23	44.72	0.24		51.30	30.00	55.05	7.31%	1.40	55.05	1.56	
	3	56.30	5.00	56.32	0.03%	0.14	56.32	0.16		56.80	18.00	58.41	2.83%	0.24	58.41	0.29	
	4	50.30	2.00	50.26	-0.08%	0.17	50.26	0.26		50.70	6.00	50.66	-0.07%	0.18	50.66	0.23	
	5	49.90	43.00	49.93	0.06%	0.17	49.93	0.24		53.80	38.00	57.41	6.71%	0.87	57.41	1.01	
	6	49.50	1.00	49.52	0.03%	0.19	49.52	0.34		53.30	11.00	52.08	-2.29%	0.50	52.08	0.56	
	7	64.60	1.00	64.57	-0.04%	0.25	64.57	0.33		68.20	26.00	64.57	-5.32%	0.33	64.57	0.50	
	8	40.50	0.50	40.48	-0.06%	0.24	40.48	0.33		40.50	3.00	40.85	0.88%	0.23	40.85	0.27	
3																	
	1	37.40	1.00	37.45	0.13%	0.19	37.45	0.22		37.40	0.50	37.45	0.13%	0.17	37.45	0.19	
	2	37.40	5.00	37.38	-0.04%	0.15	37.38	0.18		38.30	15.00	37.61	-1.81%	0.18	37.61	0.24	
	3	41.00	4.00	40.96	-0.09%	0.19	40.96	0.25		41.00	5.00	40.96	-0.09%	0.19	40.96	0.27	
	4	43.40	3.00	43.37	-0.08%	0.20	43.37	0.25		43.40	13.00	43.37	-0.08%	0.18	43.37	0.23	
	5	38.80	0.50	38.79	-0.02%	0.14	38.79	0.17		40.80	5.00	43.46	6.53%	0.24	43.66	0.24	
	6	41.30	2.00	41.33	0.08%	0.15	41.33	0.21		41.30	5.00	41.33	0.08%	0.18	41.33	0.22	
	7	44.40	2.00	44.38	-0.04%	0.14	44.38	0.17		44.40	6.00	44.97	1.28%	0.15	44.97	0.18	
	8	46.50	2.00	46.48	-0.03%	0.13	46.48	0.17		46.50	7.00	46.48	-0.03%	0.15	46.48	0.24	
4																	
	1	57.30	2.00	57.27	-0.06%	0.18	57.27	0.21		57.30	2.00	61.22	6.84%	0.38	61.22	0.43	
	2	47.30	3.00	47.26	-0.08%	0.27	47.26	0.32		49.30	29.00	49.97	1.36%	0.86	49.97	0.98	
	3	46.90	0.50	46.90	0.00%	0.15	46.90	0.19		46.90	1.00	46.90	0.00%	0.16	46.90	0.20	
	4	53.30	0.50	53.34	0.07%	0.18	53.34	0.21		53.30	1.00	53.34	0.07%	0.15	53.34	0.18	
	5	44.70	7.00	44.70	0.00%	0.19	44.70	0.21		44.70	9.00	48.77	9.12%	1.24	48.77	1.29	
	6	50.20	3.00	50.21	0.02%	0.15	50.21	0.18		52.20	30.00	52.22	0.04%	0.25	52.22	0.33	
	7	57.20	0.50	57.15	-0.09%	0.25	57.15	0.28		59.90	1.00	57.61	-3.82%	1.39	57.61	1.48	
	8	50.10	1.00	50.07	-0.06%	0.15	50.07	0.21		50.10	1.00	50.07	-0.06%	0.16	50.07	0.24	
	Total	**1,476.60**	**3.73**	**1,476.46**	**-0.01%**	**0.18**	**1,476.46**	**0.23**		**1,520.00**	**23.64**	**1,539.53**	**1.28%**	**0.36**	**1,539.71**	**0.42**	

Table II.12: Pollaris et al. (2016) Instances with 15 customers

		without Axle Weight Constraint							with Axle Weight Constraint						
		best				avg.			best					avg.	
		Pollaris et al. (2016)		ALNSxDBLF			ALNSxDBLF		Pollaris et al. (2016)		ALNSxDBLF			ALNSxDBLF	
Class	No	*ttd*	*time* [s]	*ttd*	diff. *ttd*	*time* [s]	*ttd*	*time* [s]	*ttd*	*time* [s]	*ttd*	diff. *ttd*	*time* [s]	*ttd*	*time* [s]
1															
	1	59.30	30.00	59.25	-0.08%	0.38	59.25	0.44			69.41		0.77	69.50	0.61
	2	54.60	5.00	54.57	-0.06%	0.22	54.57	0.30			75.16		0.43	75.16	0.51
	3	62.10	125.00	62.15	0.07%	0.32	62.15	0.44	62.10	481.00	62.92	1.32%	0.35	62.92	0.43
	4	54.10	97.00	54.11	0.02%	0.29	54.11	0.42	55.10	1,252.00	55.16	0.12%	0.29	55.16	0.35
	5	56.60	19.00	56.63	0.05%	0.27	56.63	0.40	58.40	96.00	59.88	2.53%	0.36	59.88	0.43
	6	64.10	56.00	64.08	-0.04%	0.37	64.08	0.48	68.50	4,735.00	73.22	6.89%	0.27	73.22	0.39
	7	62.50	144.00	62.54	0.06%	0.36	62.54	0.45	63.60	1,723.00	63.58	-0.04%	0.40	63.58	0.45
	8	56.20	12.00	56.20	0.00%	0.50	56.20	0.55	58.60	800.00	58.62	0.03%	0.35	58.62	0.46
2															
	1	56.60	10.00	56.57	-0.05%	0.55	56.57	0.62	59.70	230.00	62.28	4.33%	0.49	62.28	0.57
	3	60.30	97.00	60.32	0.03%	0.34	60.32	0.42	64.00	664.00	62.76	-1.94%	1.80	62.76	1.93
	4	58.70	16.00	58.70	0.00%	0.28	58.70	0.36	58.70	28.00	64.11	9.21%	0.48	64.11	0.56
	5	62.00	15.00	62.02	0.03%	0.60	62.02	0.63	75.40	56.00	63.73	-15.47%	1.59	63.73	1.69
	6	59.50	94.00	59.45	-0.08%	0.37	59.45	0.48			63.67		1.46	63.67	1.59
	7	68.80	1,930.00	68.77	-0.04%	0.49	68.77	0.65	68.80	1,632.00	70.01	1.75%	0.61	70.01	0.68
	8	65.80	280.00	65.80	0.00%	0.53	65.80	0.57	68.20	193.00	71.47	4.79%	0.70	71.47	0.77
3															
	1	56.20	280.00	56.20	0.00%	0.34	56.20	0.38	56.20	726.00	56.20	0.00%	0.39	56.20	0.43
	2	51.50	83.00	51.52	0.05%	0.27	51.52	0.32	51.50	316.00	51.85	0.67%	0.27	51.85	0.30
	3	53.30	29.00	53.31	0.02%	0.28	53.31	0.40	54.70	265.00	53.31	-2.54%	0.34	53.31	0.40
	4	57.80	15.00	57.83	0.05%	0.33	57.83	0.36	57.80	23.00	57.83	0.05%	0.35	57.83	0.54
	5	55.60	20.00	55.62	0.04%	0.30	55.62	0.38	56.70	52.00	55.62	-1.90%	0.33	55.62	0.41
	6	58.00	89.00	58.00	0.00%	0.29	58.00	0.33	58.00	213.00	58.00	0.00%	0.28	58.00	0.33
	7	59.00	58.00	59.04	0.06%	0.32	59.04	0.34	59.00	70.00	59.04	0.06%	0.31	59.04	0.34
	8	46.40	40.00	46.44	0.09%	0.26	46.44	0.28	47.10	380.00	46.44	-1.40%	0.31	46.44	0.39
4															
	1	81.70	6.00	81.68	-0.02%	0.46	81.68	0.49	93.20	62.00	91.74	-1.57%	1.51	91.74	1.62
	2	79.70	66.00	79.70	0.01%	0.57	79.70	0.60	83.40	125.00	90.41	8.40%	1.87	90.41	1.99
	3	77.90	56.00	77.86	-0.05%	0.53	77.86	0.58	78.40	84.00	80.96	3.26%	1.16	80.96	1.37
	4	88.00	54.00	87.99	-0.01%	0.64	87.99	0.78	88.00	23.00	87.99	-0.01%	0.50	87.99	0.61
	5	65.40	46.00	65.36	-0.07%	0.47	65.36	0.55	65.40	55.00	66.12	1.09%	0.60	66.12	0.71
	6	74.70	32.00	74.71	0.01%	0.31	74.71	0.41	74.70	23.00	79.99	7.09%	1.58	79.99	1.66
	7	63.90	206.00	63.94	0.07%	0.41	63.94	0.52	64.00	132.00	64.79	1.23%	0.55	64.79	0.62
	8	63.30	38.00	63.34	0.06%	0.46	63.34	0.55	63.30	80.00	63.59	0.45%	0.40	63.59	0.45
	Total	**1,760.20**	**139.96**	**1,760.43**	**0.01%**	**0.40**	**1,760.43**	**0.47**	**1,812.50**	**518.54**	**1,831.61**	**1.05%**	**0.66**	**1,831.62**	**0.75**

Table II.13: Pollaris et al. (2017) Instances with 100 customers

		without Axle Weight Constraint								with Axle Weight Constraint							
		best				avg.				best					avg.		
		Pollaris et al. (2016)		ALNSxDBLF				ALNSxDBLF		Pollaris et al. (2016)		ALNSxDBLF			ALNSxDBLF		
Class	No	*ttd*	*time* [s]	*ttd*	diff. *ttd*	*time* [s]		*ttd*	*time* [s]	*ttd*	*time* [s]	*ttd*	diff. *ttd*	*time* [s]	*ttd*	*time* [s]	
1																	
	1	74.00	29.00	73.99	-0.02%	0.63		73.99	0.64	74.20	345.00	78.56	5.87%	0.74	78.72	0.90	
	3	72.60	731.00	72.57	-0.04%	0.66		72.57	0.82			75.53		0.63	75.53	0.73	
	4	72.00	311.00	72.01	0.01%	0.58		72.01	0.69	72.00	1,647.00	72.01	0.01%	0.56	72.01	0.60	
	5	70.80	5,917.00	70.76	-0.06%	0.73		70.76	0.84			75.45		0.81	75.45	0.91	
	6	61.70	21.00	61.75	0.08%	0.57		61.75	0.66			64.35		0.51	64.35	0.66	
	7	69.80	25.00	69.80	0.00%	0.53		69.80	0.58	69.80	55.00	69.80	0.00%	0.53	69.80	0.65	
	8	68.50	2,165.00	68.50	-0.01%	0.70		68.50	0.74			76.68		0.94	76.68	1.10	
2																	
	1	91.70	728.00	91.67	-0.04%	0.84		91.67	1.01			97.59		0.86	97.59	0.90	
	2	87.80	749.00	87.77	-0.03%	1.19		87.77	1.29	87.80	1,613.00	89.22	1.61%	0.74	89.22	0.78	
	3	113.00	92.00	113.04	0.03%	1.30		113.04	1.62			139.11		3.64	139.11	3.89	
	4	99.50	308.00	99.54	0.04%	0.85		99.54	1.03	101.40	1,116.00	116.78	15.16%	3.11	116.78	3.27	
	5	94.40	281.00	94.44	0.04%	1.26		94.44	1.33			105.41		2.98	105.41	3.05	
	6	104.60	1,420.00	104.60	0.00%	1.03		104.60	1.11	111.30	4,421.00	133.39	19.85%	3.64	133.39	4.14	
	7	90.80	64.00	90.80	0.00%	0.90		90.80	1.04	93.50	1,305.00	96.78	3.51%	2.14	97.87	2.52	
	8	91.40	624.00	91.35	-0.05%	1.00		91.35	1.11	91.40	4,073.00	109.53	19.83%	2.57	109.53	2.83	
3																	
	1	67.30	213.00	67.30	0.01%	0.58		67.30	0.82	67.30	898.00	67.71	0.61%	0.65	67.71	0.67	
	2	68.50	668.00	68.52	0.03%	0.65		68.52	0.79	68.50	1,553.00	68.52	0.03%	0.63	68.52	0.66	
	3	78.70	1,139.00	78.72	0.03%	0.67		78.72	0.75	78.80	4,353.00	79.85	1.33%	0.65	79.85	0.67	
	4	63.10	5,458.00	63.05	-0.08%	0.62		63.05	0.68	63.10	2,360.00	63.05	-0.08%	0.64	63.05	0.72	
	5	68.30	1,000.00	68.28	-0.03%	0.69		68.28	0.83	68.30	6,936.00	68.28	-0.03%	0.75	68.28	0.90	
	6	78.40	1,035.00	78.42	0.02%	0.63		78.42	0.74	78.40	1,450.00	78.55	0.20%	0.59	78.55	0.66	
	7	63.60	204.00	63.62	0.02%	0.86		63.62	0.91	63.60	4,071.00	63.62	0.02%	0.80	63.62	0.83	
	8	67.00	2,555.00	66.97	-0.05%	0.63		66.97	0.85			66.97		0.76	66.97	0.87	
4																	
	1	80.90	3,065.00	80.88	-0.02%	0.58		80.88	0.64			84.14		0.64	84.14	0.77	
	2	59.50	348.00	59.53	0.05%	0.73		59.53	0.80			60.80		0.91	60.80	1.01	
	3	88.90	987.00	88.92	0.02%	0.79		88.92	0.82	88.90	1,416.00	89.16	0.29%	0.66	89.16	0.73	
	4	86.60	90.00	86.61	0.01%	1.06		86.61	1.15	89.30	365.00	91.93	2.95%	2.73	91.93	2.94	
	5	100.30	216.00	100.34	0.04%	1.18		100.34	1.39	100.30	1,322.00	102.91	2.60%	0.91	102.91	1.07	
	6	122.70	129.00	122.65	-0.04%	0.63		122.65	0.72	127.60	660.00	130.99	2.66%	3.58	130.99	3.77	
	7	92.60	149.00	92.65	0.05%	0.79		92.65	0.88	97.50	1,838.00	103.67	6.33%	1.13	103.67	1.33	
	8	89.90	37.00	89.84	-0.06%	0.62		89.84	0.83	89.90	83.00	98.72	9.81%	2.61	98.72	2.71	
	Total	**2,538.90**	**992.19**	**2,538.87**	**0.00%**	**0.79**		**2,538.89**	**0.91**	**1,782.90**	**1,994.29**	**2,719.05**	**52.51%**	**1.39**	**2,720.31**	**1.52**	

Table II.14: Pollaris et al. (2016) Instances with 25 customers

		without Axle Weight Constraint				**with Axle Weight Constraint**			
		best		**avg.**		**best**		**avg.**	
		ALNSxDBLF		**ALNSxDBLF**		**ALNSxDBLF**		**ALNSxDBLF**	
Class	**No**	*ttd*	*time* [s]	*ttd*	*time* [s]	*ttd*	*time* [s]	*ttd*	*time* [s]
1									
	1	84.55	1.00	84.55	1.19	90.71	1.66	90.96	1.79
	2	78.23	1.07	78.23	1.12	90.49	2.70	91.16	2.91
	3	77.98	0.75	77.98	0.91	84.04	1.56	84.04	1.81
	4	90.03	1.01	90.03	1.25	92.81	1.32	92.81	1.53
	5	85.80	1.09	85.80	1.35	107.00	2.24	107.36	2.59
	6	93.42	1.17	93.42	1.32	100.43	1.14	100.43	1.24
	7	83.05	1.23	83.05	1.29	104.06	3.59	106.88	2.71
	8	81.57	0.95	81.57	1.01	86.26	1.14	86.26	1.20
2									
	1	122.41	1.60	122.41	1.72	163.62	4.64	163.62	4.91
	2	111.38	1.53	111.38	2.16	130.39	4.79	130.39	4.88
	3	90.00	1.38	90.00	1.64	122.51	5.14	122.51	5.35
	4	168.44	1.82	168.44	1.97	172.04	4.66	172.04	4.85
	5	114.44	1.36	114.44	1.42	138.95	4.78	138.95	5.02
	6	101.66	1.22	101.66	1.34	119.05	4.56	119.05	4.73
	7	120.18	1.18	120.18	1.34	146.59	2.89	146.59	4.22
	8	109.27	1.60	109.27	1.77	124.92	2.39	124.92	3.09
3									
	1	90.22	1.13	90.22	1.48	91.82	1.54	91.82	1.69
	2	73.45	1.28	73.45	1.63	73.69	1.26	73.69	1.37
	3	79.42	1.08	79.42	1.09	79.42	0.98	79.42	1.09
	4	79.92	1.03	79.92	1.10	80.48	1.30	80.48	1.48
	5	82.47	1.15	82.47	1.22	82.47	1.10	82.47	1.23
	6	83.65	1.01	83.65	1.20	85.47	1.28	85.47	1.43
	7	81.68	1.24	81.68	1.32	82.79	1.43	82.79	1.68
	8	83.48	1.06	83.48	1.24	83.48	1.07	83.48	1.30
4									
	1	125.07	1.49	125.07	1.72	136.12	4.14	136.12	4.40
	2	85.68	1.44	85.68	1.53	99.47	2.76	99.75	3.35
	3	120.76	2.71	120.76	2.85	132.12	4.01	132.12	4.30
	4	115.85	1.67	115.85	1.87	120.66	4.40	120.66	4.48
	5	104.86	1.59	104.86	1.66	104.88	1.07	104.88	1.22
	6	99.64	1.78	99.64	2.08	122.03	5.09	122.03	5.22
	7	120.78	1.65	120.78	1.86	132.77	5.39	132.77	5.44
	8	102.74	1.90	102.74	1.96	109.36	4.42	109.36	4.56
	Total	**3,142.09**	**1.35**	**3,142.08**	**1.52**	**3,490.88**	**2.83**	**3,495.28**	**3.03**

Table II.15: Pollaris et al. (2017) Instances with 50 customers

		without Axle Weight Constraint								with Axle Weight Constraint							
		best				avg.				best				avg.			
		ILS	ALNSxDBLF			ILS	ALNSxDBLF			ILS	ALNSxDBLF			ILS	ALNSxDBLF		
Class	No	*ttd*	*ttd*	diff. *ttd*	*time* [s]	*ttd*	*ttd*	diff. *ttd*	*time* [s]	*ttd*	*ttd*	diff. *ttd*	*time* [s]	*ttd*	*ttd*	diff. *ttd*	*time* [s]
1																	
	1	152.20	152.22	0.01%	9.61	152.60	152.58	-0.01%	10.45	154.10	158.13	2.61%	12.07	154.60	158.13	2.28%	13.15
	2	133.80	133.82	0.02%	11.01	134.10	134.34	0.18%	10.79	143.30	146.60	2.30%	16.26	145.30	146.99	1.16%	15.99
	3	145.40	145.26	-0.10%	12.18	145.60	145.39	-0.14%	10.54	147.70	148.47	0.52%	12.85	148.00	148.81	0.55%	13.10
	4	150.80	150.79	-0.01%	10.00	150.90	150.79	-0.07%	10.50	152.10	156.21	2.70%	15.67	154.60	158.29	2.39%	14.14
	5	166.10	166.07	-0.02%	6.41	166.20	166.07	-0.08%	7.31	168.20	170.04	1.09%	10.59	170.20	170.38	0.11%	11.71
	6	153.20	153.23	0.02%	6.22	153.30	153.23	-0.05%	7.92	156.20	159.43	2.07%	13.19	156.70	159.74	1.94%	13.59
	7	142.50	142.21	-0.20%	14.71	143.40	142.79	-0.43%	10.79	146.30	147.33	0.70%	16.35	148.20	148.62	0.28%	15.86
	8	142.30	142.31	0.01%	7.63	145.30	142.64	-1.83%	9.58	147.50	154.27	4.59%	11.55	149.10	154.48	3.61%	12.19
2																	
	1	170.30	170.32	0.01%	5.52	170.30	170.32	0.01%	6.81	177.50	222.06	25.10%	17.47	178.60	222.06	24.33%	17.68
	2	200.50	200.52	0.01%	5.39	200.70	200.52	-0.09%	6.67	203.80	256.53	25.88%	17.85	207.30	256.53	23.75%	17.99
	3	188.40	188.44	0.02%	6.04	188.90	188.44	-0.24%	8.35	192.80	231.23	19.93%	18.77	197.30	231.23	17.20%	18.88
	4	183.00	183.55	0.30%	6.42	183.30	183.55	0.14%	7.70	187.00	222.76	19.13%	14.46	188.80	222.76	17.99%	15.13
	5	190.90	190.86	-0.02%	6.29	191.50	191.09	-0.21%	7.40	194.50	206.50	6.17%	11.41	196.50	206.50	5.09%	11.67
	6	168.40	168.37	-0.02%	5.41	168.40	168.37	-0.02%	6.01	168.40	176.19	4.63%	15.57	169.50	176.61	4.19%	15.48
	7	184.20	184.23	0.02%	5.87	184.40	184.23	-0.09%	7.00	189.40	229.49	21.17%	18.03	195.30	229.49	17.51%	18.64
	8	192.50	192.48	-0.01%	5.94	192.70	192.48	-0.11%	6.78	199.60	245.32	22.91%	14.25	203.00	245.32	20.85%	15.33
3																	
	1	143.90	143.91	0.00%	6.99	144.10	143.91	-0.13%	8.01	144.30	144.26	-0.02%	8.13	144.80	144.26	-0.37%	9.36
	2	156.90	156.92	0.01%	8.12	157.10	156.92	-0.11%	9.70	157.30	157.59	0.18%	8.54	158.40	157.59	-0.51%	10.20
	3	143.80	143.84	0.03%	7.39	144.00	143.84	-0.11%	8.30	144.30	144.25	-0.03%	10.34	146.70	144.25	-1.67%	10.91
	4	149.10	149.41	0.21%	9.66	149.20	149.41	0.14%	11.04	149.10	149.73	0.42%	12.50	150.40	149.81	-0.39%	13.23
	5	158.30	158.35	0.03%	8.94	158.50	158.35	-0.09%	9.76	159.80	160.37	0.36%	10.49	160.80	160.39	-0.25%	11.25
	6	143.30	143.30	0.00%	9.00	143.90	143.29	-0.42%	9.65	144.30	143.61	-0.48%	9.64	145.10	143.61	-1.03%	10.93
	7	146.50	146.71	0.14%	7.97	147.30	146.71	-0.40%	8.98	146.60	147.72	0.76%	10.83	148.00	147.85	-0.10%	11.65
	8	130.00	129.96	-0.03%	10.81	130.00	130.30	0.23%	12.18	130.30	130.14	-0.13%	8.89	131.20	130.19	-0.77%	10.41
4																	
	1	181.40	181.36	-0.02%	7.23	182.30	181.36	-0.52%	8.62	184.60	212.03	14.86%	15.40	185.50	212.03	14.30%	15.73
	2	210.00	209.98	-0.01%	5.64	210.10	209.98	-0.06%	6.38	210.70	244.79	16.18%	13.23	210.80	244.79	16.12%	13.33
	3	197.50	197.32	-0.09%	6.12	198.70	197.38	-0.66%	7.79	199.00	207.54	4.29%	12.41	202.00	210.44	4.18%	12.60
	4	193.70	193.66	-0.02%	5.43	193.70	193.66	-0.02%	6.21	199.50	210.79	5.66%	13.03	200.40	212.62	6.10%	11.14
	5	198.10	198.10	0.00%	5.06	199.10	198.10	-0.50%	5.97	205.50	215.68	4.95%	11.54	209.00	215.68	3.20%	11.80
	6	212.50	212.45	-0.02%	10.13	212.90	212.75	-0.07%	8.15	212.80	215.24	1.15%	9.31	213.40	216.21	1.32%	9.16
	7	199.40	199.41	0.01%	5.48	199.50	199.41	-0.05%	6.42	202.90	215.09	6.01%	9.92	203.60	215.08	5.64%	12.40
	8	222.00	221.97	-0.01%	3.65	222.00	221.97	-0.01%	5.67	234.30	247.36	5.57%	12.33	237.20	247.35	4.28%	12.49
	Total	**5,450.90**	**5,451.33**	**0.01%**	**7.57**	**5,464.00**	**5,454.17**	**-0.18%**	**8.36**	**5,553.70**	**5,976.75**	**7.62%**	**12.90**	**5,610.30**	**5,988.09**	**6.73%**	**13.35**

Table II.16: Pollaris et al. (2017) Instances with 75 customers

		without Axle Weight Constraint								with Axle Weight Constraint							
		best				avg.				best				avg.			
		ILS	ALNSxDBLF			ILS	ALNSxDBLF			ILS	ALNSxDBLF			ILS	ALNSxDBLF		
Class	No	*ttd*	*ttd*	diff. *ttd*	*time* [s]	*ttd*	*ttd*	diff. *ttd*	*time* [s]	*ttd*	*ttd*	diff. *ttd*	*time* [s]	*ttd*	*ttd*	diff. *ttd*	*time* [s]
1	1	201.10	201.36	0.13%	37.23	201.70	201.77	0.03%	35.68	203.60	205.40	0.88%	33.31	207.90	205.54	-1.14%	39.45
	2	206.30	206.32	0.01%	32.20	208.00	206.39	-0.77%	32.37	213.60	216.77	1.48%	45.47	216.20	218.31	0.98%	46.83
	3	207.80	207.82	0.01%	26.80	208.10	207.82	-0.13%	28.59	213.30	218.40	2.39%	49.69	215.90	222.16	2.90%	44.30
	4	212.70	211.12	-0.74%	36.82	213.00	212.45	-0.26%	32.76	217.50	216.42	-0.49%	50.96	219.50	217.74	-0.80%	48.85
	5	199.70	198.89	-0.41%	32.97	200.20	198.98	-0.61%	32.65	201.80	205.52	1.84%	49.78	205.40	206.27	0.42%	45.53
	6	198.70	198.68	-0.01%	28.88	200.00	198.96	-0.52%	31.33	202.80	203.29	0.24%	52.47	208.30	204.97	-1.60%	47.96
	7	213.20	213.39	0.09%	40.03	213.90	213.45	-0.21%	31.76	219.70	221.30	0.73%	53.04	223.60	222.31	-0.58%	51.38
	8	206.40	206.40	0.00%	28.41	207.30	206.59	-0.34%	31.16	212.40	217.92	2.60%	50.10	215.00	221.59	3.07%	47.95
2	1	273.80	273.75	-0.02%	15.70	273.80	273.75	-0.02%	20.30	283.50	341.93	20.61%	37.16	288.40	341.93	18.56%	38.23
	2	260.80	260.82	0.01%	21.74	260.90	260.82	-0.03%	22.63	262.80	279.70	6.43%	36.02	266.60	282.06	5.80%	37.65
	3	298.90	297.58	-0.44%	25.16	300.00	298.72	-0.43%	24.23	308.20	384.06	24.61%	42.17	312.80	384.06	22.78%	42.61
	4	275.10	274.50	-0.22%	25.50	275.80	275.08	-0.26%	25.95	282.50	319.97	13.26%	38.96	285.80	319.97	11.96%	39.87
	5	323.70	323.72	0.01%	16.14	323.90	323.72	-0.06%	19.15	329.20	351.02	6.63%	30.99	332.50	351.02	5.57%	31.69
	6	262.80	262.00	-0.30%	35.25	263.20	262.30	-0.34%	30.23	264.30	289.13	9.40%	45.81	267.20	290.70	8.79%	44.05
	7	314.90	314.82	-0.02%	24.43	315.30	314.89	-0.13%	24.97	323.70	396.39	22.46%	40.17	327.90	396.39	20.89%	40.38
	8	325.90	325.75	-0.05%	19.16	326.10	325.95	-0.05%	23.27	333.30	366.01	9.81%	36.35	334.70	366.01	9.35%	36.51
3	1	220.20	220.81	0.28%	36.71	221.90	221.15	-0.34%	37.53	223.70	221.09	-1.16%	44.77	225.50	222.02	-1.54%	41.32
	2	194.50	194.50	0.00%	35.59	195.30	194.84	-0.24%	36.81	196.00	194.53	-0.75%	37.93	198.10	195.76	-1.18%	36.50
	3	201.40	201.41	0.01%	24.63	201.50	201.49	0.00%	26.50	203.30	202.86	-0.22%	33.74	206.60	202.99	-1.75%	36.90
	4	203.70	203.65	-0.02%	35.76	204.80	203.87	-0.45%	35.00	204.80	203.68	-0.55%	36.52	209.10	203.88	-2.50%	33.53
	5	204.30	204.28	-0.01%	26.29	205.00	204.40	-0.29%	29.41	205.60	204.76	-0.41%	37.78	207.80	205.05	-1.32%	37.82
	6	229.60	228.44	-0.51%	40.53	229.80	229.61	-0.08%	32.87	230.00	228.44	-0.68%	35.36	232.20	229.53	-1.15%	35.46
	7	202.00	202.33	0.16%	29.52	206.60	202.43	-2.02%	30.36	202.00	201.57	-0.21%	36.21	204.60	202.26	-1.14%	36.29
	8	209.60	209.44	-0.07%	39.97	211.60	210.66	-0.44%	34.80	212.50	209.78	-1.28%	42.94	215.40	210.85	-2.11%	39.32
4	1	337.30	337.19	-0.03%	22.06	337.80	337.25	-0.16%	23.39	340.60	374.14	9.85%	31.88	342.30	374.14	9.30%	32.30
	2	326.70	326.67	-0.01%	19.92	326.80	326.67	-0.04%	23.75	331.90	349.15	5.20%	30.10	335.40	349.15	4.10%	30.86
	3	327.00	327.00	0.00%	13.71	328.20	327.00	-0.37%	17.61	332.60	355.82	6.98%	28.44	334.20	355.82	6.47%	29.44
	4	322.50	322.13	-0.12%	24.86	325.10	322.13	-0.91%	26.98	326.80	352.94	8.00%	33.56	330.90	352.94	6.66%	35.43
	5	270.50	270.52	0.01%	20.12	271.60	270.52	-0.40%	25.92	271.70	276.78	1.87%	32.53	274.30	276.91	0.95%	33.36
	6	286.40	286.42	0.01%	17.54	287.00	286.89	-0.04%	22.88	287.80	288.84	0.36%	26.47	290.10	289.02	-0.37%	29.87
	7	287.80	287.81	0.00%	20.39	288.20	287.89	-0.11%	20.86	295.70	322.81	9.17%	32.13	297.80	323.93	8.77%	30.38
	8	268.60	268.46	-0.05%	21.97	269.20	268.57	-0.23%	20.85	270.30	271.84	0.57%	34.90	273.50	274.89	0.51%	31.01
	Total	**8,073.90**	**8,067.98**	**-0.07%**	**27.37**	**8,101.60**	**8,077.01**	**-0.30%**	**27.89**	**8,207.50**	**8,692.28**	**5.91%**	**38.99**	**8,305.50**	**8,720.17**	**4.99%**	**38.53**

Table II.17: Pollaris et al. (2017) Instances with 100 customers

		without Axle Weight Constraint								with Axle Weight Constraint							
		best				avg.				best				avg.			
		ILS	ALNSxDBLF			ILS	ALNSxDBLF			ILS	ALNSxDBLF			ILS	ALNSxDBLF		
Class	No	*ttd*	*ttd*	diff. *ttd*	*time* [s]	*ttd*	*ttd*	diff. *ttd*	*time* [s]	*ttd*	*ttd*	diff. *ttd*	*time* [s]	*ttd*	*ttd*	diff. *ttd*	*time* [s]
1																	
	1	271.00	270.54	-0.17%	71.52	272.70	271.40	-0.48%	72.15	278.00	281.76	1.35%	109.49	282.60	284.05	0.51%	107.48
	2	280.60	282.17	0.56%	56.10	282.20	282.21	0.00%	68.29	289.10	294.02	1.70%	112.94	292.90	295.59	0.92%	97.74
	3	254.30	253.69	-0.24%	65.48	256.60	253.71	-1.13%	74.31	265.00	266.23	0.46%	123.68	268.50	270.70	0.82%	112.24
	4	273.10	273.13	0.01%	84.50	275.30	273.49	-0.66%	80.46	285.80	299.60	4.83%	91.84	291.80	300.83	3.09%	100.46
	5	260.10	259.92	-0.07%	78.73	260.90	260.00	-0.34%	80.37	266.40	268.71	0.87%	101.61	272.20	272.55	0.13%	88.31
	6	274.40	273.49	-0.33%	80.72	275.30	274.09	-0.44%	74.17	285.40	290.14	1.66%	91.12	289.40	293.59	1.45%	89.74
	7	274.20	274.16	-0.02%	66.53	275.10	274.25	-0.31%	71.58	287.60	294.26	2.31%	84.80	292.50	299.33	2.34%	95.41
	8	271.20	271.25	0.02%	60.70	272.20	271.35	-0.31%	66.07	280.50	278.22	-0.81%	110.77	287.00	279.09	-2.76%	109.59
2																	
	1	387.70	386.47	-0.32%	43.77	388.60	386.69	-0.49%	49.30	397.30	459.00	15.53%	86.47	402.20	459.00	14.12%	88.32
	2	360.30	360.35	0.01%	48.69	362.30	360.99	-0.36%	54.51	365.70	413.13	12.97%	80.65	369.90	413.13	11.69%	81.94
	3	355.40	354.77	-0.18%	44.72	355.90	354.78	-0.31%	49.82	363.90	413.58	13.65%	94.35	371.50	413.58	11.33%	96.22
	4	382.30	382.35	0.01%	46.51	383.80	382.45	-0.35%	55.45	388.90	478.07	22.93%	99.01	397.30	478.07	20.33%	99.72
	5	364.00	363.95	-0.01%	53.15	364.90	363.95	-0.26%	55.29	370.80	457.28	23.32%	112.74	377.90	457.28	21.01%	114.05
	6	331.30	331.31	0.00%	48.53	332.80	331.34	-0.44%	51.24	337.70	426.30	26.24%	110.43	341.40	426.30	24.87%	111.60
	7	381.50	381.45	-0.01%	45.36	382.30	381.62	-0.18%	53.54	390.30	477.58	22.36%	106.54	397.40	477.58	20.18%	108.00
	8	383.00	382.80	-0.05%	56.49	384.20	382.95	-0.33%	54.40	387.60	420.13	8.39%	94.39	393.50	429.86	9.24%	92.27
3																	
	1	248.30	246.06	-0.90%	76.95	249.50	246.73	-1.11%	75.99	253.50	248.24	-2.07%	77.21	258.90	248.58	-3.99%	89.14
	2	266.40	264.68	-0.64%	76.54	267.30	265.28	-0.76%	77.91	270.30	265.71	-1.70%	78.86	274.30	265.97	-3.04%	86.14
	3	270.80	270.61	-0.07%	83.20	273.40	271.22	-0.80%	77.47	277.20	271.80	-1.95%	97.48	280.20	272.45	-2.77%	91.09
	4	252.90	251.54	-0.54%	65.01	253.90	251.58	-0.91%	72.46	259.20	251.94	-2.80%	77.33	261.20	252.64	-3.28%	92.26
	5	266.00	263.94	-0.78%	73.74	266.80	266.46	-0.13%	64.74	266.70	266.11	-0.22%	71.13	271.50	268.55	-1.09%	52.09
	6	276.50	275.60	-0.33%	73.74	277.90	276.55	-0.49%	80.16	281.10	275.79	-1.89%	83.10	285.10	276.69	-2.95%	82.25
	7	259.80	257.73	-0.80%	68.08	261.50	258.15	-1.28%	71.82	263.10	258.65	-1.69%	72.06	268.30	259.59	-3.25%	77.04
	8	265.40	265.32	-0.03%	74.90	266.50	265.33	-0.44%	73.98	266.30	265.34	-0.36%	85.65	271.10	265.48	-2.07%	96.93
4																	
	1	391.40	391.34	-0.02%	58.58	393.30	391.68	-0.41%	58.31	398.30	451.42	13.34%	75.35	401.90	451.42	12.32%	75.96
	2	410.50	410.54	0.01%	45.08	412.00	410.65	-0.33%	51.84	421.50	460.57	9.27%	64.20	426.10	460.57	8.09%	65.79
	3	429.40	429.40	0.00%	34.69	431.20	429.61	-0.37%	38.71	437.00	450.19	3.02%	55.29	443.80	452.85	2.04%	59.74
	4	354.90	354.89	0.00%	52.65	357.20	355.18	-0.57%	52.06	363.80	398.14	9.44%	74.82	368.00	401.09	8.99%	68.72
	5	353.30	353.31	0.00%	50.44	354.20	353.37	-0.23%	53.36	361.70	388.05	7.28%	68.84	366.70	388.05	5.82%	69.55
	6	363.20	362.10	-0.30%	56.52	364.20	362.33	-0.51%	56.91	375.40	411.23	9.54%	72.66	379.30	411.23	8.42%	73.42
	7	390.80	390.44	-0.09%	61.95	392.60	391.12	-0.38%	59.08	401.20	446.68	11.34%	70.07	405.70	446.68	10.10%	70.39
	8	374.70	374.20	-0.13%	41.94	375.10	374.55	-0.15%	41.94	384.70	424.35	10.31%	55.52	390.90	424.35	8.56%	57.71
	Total	**10,278.70**	**10,263.48**	**-0.15%**	**60.80**	**10,321.70**	**10,275.06**	**-0.45%**	**63.05**	**10,521.00**	**11,352.20**	**7.90%**	**87.20**	**10,681.00**	**11,396.72**	**6.70%**	**87.54**

Paper III

Advanced loading constraints for 3D vehicle routing problems

Corinna Krebs, Jan Fabian Ehmke, Henriette Koch

Published in *OR Spectrum*, August 2021, DOI: 10.1007/s00291-021-00645-w.

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 literature. Detailed results and the implementation of loading constraints are published online.

Contents

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 is complex and requires 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 consists 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 Section 2, relevant literature is reviewed. The 3L-CVRP and the 3L-VRPTW are formulated in Section 3. In Section 4, the new definitions and the implementation variants of the loading constraints are presented. In Section 5, the hybrid algorithm is described, and Section 6 deals with the testing of the constraints. Finally, conclusions are drawn in Section 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 III.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). 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 Neighborhood 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 program 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 III.1 summarizes the approaches.

Table III.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 Neighborhood Search	Extreme Point Based First Fit	Time Limit	36
Ceschia et al. (2013)	3L-CVRP	Large-Neighborhood 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 Program 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 III.2 summarizes the related literature and highlights our contribution. As demonstrated in Table III.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.

Table III.2: Summary and Overview over Loading Constraints

References sorted by year	Geometry C1	Orthogonality C2	Rotation C3	Load Capacity C4	LIFO C5a	MLIFO C5b	Minimal Supporting Area C6a	Robust Stability C6b	Fragility C7a	Load Bearing Strength C7b	Reachability 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	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

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;

(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 the next Section 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 III.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*		

Table III.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 III.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 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.

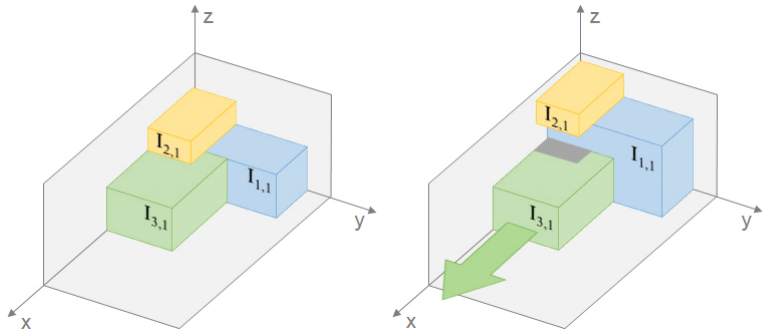
4.1.2 MLIFO (C5b)

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. III.1. For both variants, it is not allowed to place an item directly on top of another item that is delivered earlier (see Fig. III.1a). In contrast to the LIFO constraint, it is allowed that one item hangs over another item that is delivered earlier (see Fig. III.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.



(a) infeasible placements

(b) feasible placement for MLIFO, infeasible placement for LIFO

Figure III.1: Difference between LIFO and MLIFO

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. III.2, this formulation can lead to unstable, but still feasible item arrangements: When stacking several items with same density, whereby the length or width of each item enlarge by $\frac{1}{\alpha}$, an overhanging stack of items is created.

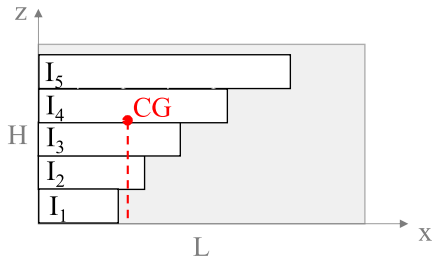


Figure III.2: Unstable, feasible stack w.r.t. Minimal Supporting Area

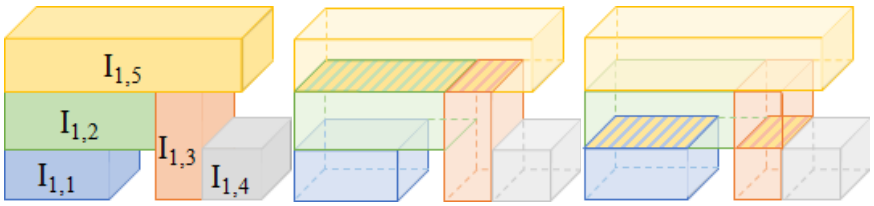
This arrangement is in accordance with the Minimal Supporting Area constraint (C6a) because for the calculation of the support for one item, only the directly underlying items are considered. According to the Science of Statics, this stack is not stable, because the x-value of the centre of gravity (CG) 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. III.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. III.3c). Each plane must obey the Minimal Supporting Area constraint. Otherwise, the constraint is violated and the placement of item $I_{i,k}$ is rejected.

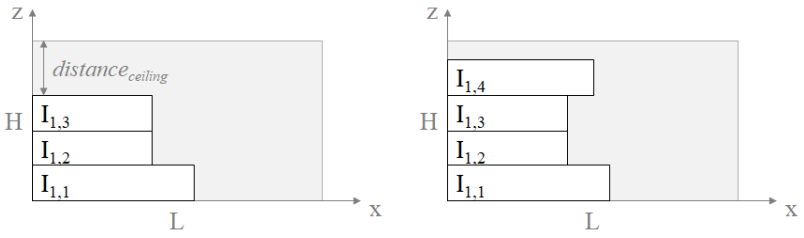


(a) item stack (b) plane 1 (c) plane 2

Figure III.3: Determination of planes for item $I_{1,5}$

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. III.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. III.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).



(a) Determination $distance_{ceiling}$ (b) Feasible Placement

Figure III.4: Implementation of Robust Stability – Top Overhanging

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 (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. III.5). Even if the non-fragile part on top of the fragile item would be infinitely small, the arrangement remains infeasible.

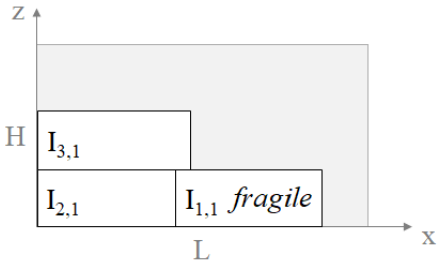


Figure III.5: Infeasible item arrangement w.r.t. Fragility constraint

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) 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:

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

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

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

The load acting on item I_u is

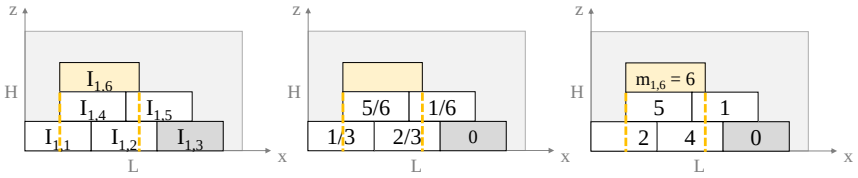
$$load_{c,u} = support_{prop} \cdot load_c, \tag{III.3}$$

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. III.6). In this approach, $load_c$ in Eq. III.3 corresponds to the

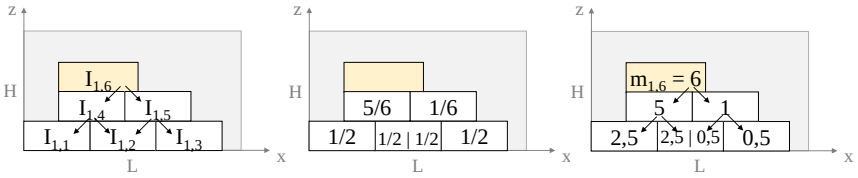
mass of I_c. The example in Fig. III.6 shows the resulting loads for the underlying items caused only by item $I_{1,6}$.



(a) Distribution (b) Support between underlying items and $I_{1,6}$ (c) Resulting load

Figure III.6: Mass Distribution according to Simplified Selection based on 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 repeated until the items on the ground are reached. Fig. III.7 shows the same exemplary situation as Fig. III.6.



(a) Distribution (b) Support between direct underlying items (c) Resulting load

Figure III.7: Mass Distribution according to Complete Selection based on item $I_{1,6}$

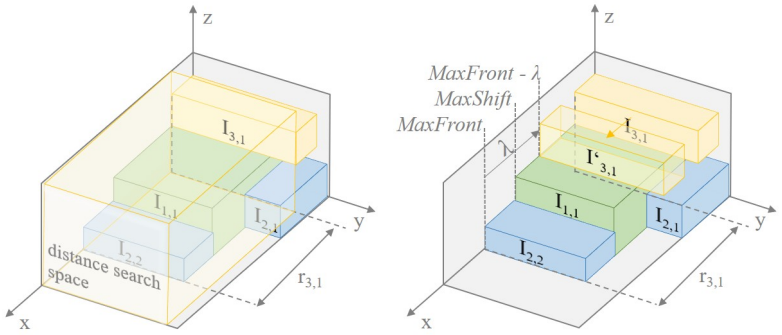
In this approach, all items of a stack contribute to the mass distribution. The resulting loads caused by item $I_{1,4}$ and item $I_{1,5}$ are calculated in the same way.

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. III.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. III.8b). Additionally, the new placement is tested w.r.t. the loading constraint set. If the item is not reachable and it cannot be shifted, the placement is rejected.

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



(a) Distance Search Space for $I_{3,1}$ (b) Shiftment of $I_{3,1}$

Figure III.8: Illustration of the Distance Search Space for $I_{3,1}$

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. III.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.

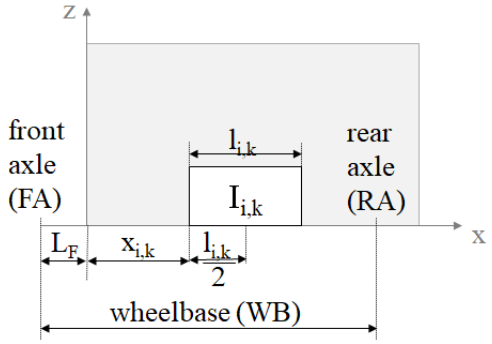


Figure III.9: Vehicle Data

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 ($g \approx 9.81 \frac{m}{s^2}$).

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

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

and

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

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

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

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

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

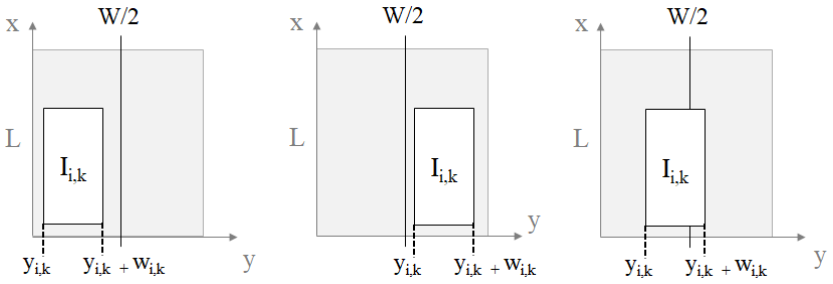
and

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

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.



(a) Left Side (b) Right Side (c) Distributed

Figure III.10: Mass Distribution according to the position of $I_{i,k}$

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. III.10a), its total mass is assigned to the left vehicle side. The same is true for the opposite right side (see Fig. III.10b). Otherwise, the mass of the item is distributed proportionally to the vehicle sides (see Fig. III.10c). The sum of all assigned masses must not exceed a certain percentage p of the load capacity D. Consequently, the following must apply:

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

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

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

Formula III.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 III.12 and in III.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. III.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 III.1 Adaptive Large Neighbourhood Search

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

1: construct initial solution s_{init}
2: $s_{best} := s_{init}$
3: $s_{curr} := s_{init}$
4: **do**
5: select removal operator rem
6: select insertion operator $inst$
7: select number of customers to be removed n_{rem}
8: determine next solution $s_{next} := inst(rem(s_{curr}, n_{rem}))$
9: check acceptance of s_{next}
10: **if** s_{next} is accepted **then**
11: $s_{curr} := s_{next}$
12: **if** $f(s_{curr}) < f(s_{best})$ **then**
13: $s_{best} := s_{next}$
14: **end if**
15: **end if**
16: **if** $iter_p$ reached **then**
17: update selection probabilities for insertion and removal heuristics
18: **end if**
19: **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 Section 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:

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

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].

5.1.5 Removal and Insertion Operators

Table III.4 shows nine removal operators and Table III.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).

Table III.4: Overview Removal Operators

Neighborhood 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 III.5: Overview Insertion Operators

Neighborhood 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

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:

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

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:

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

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. III.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:

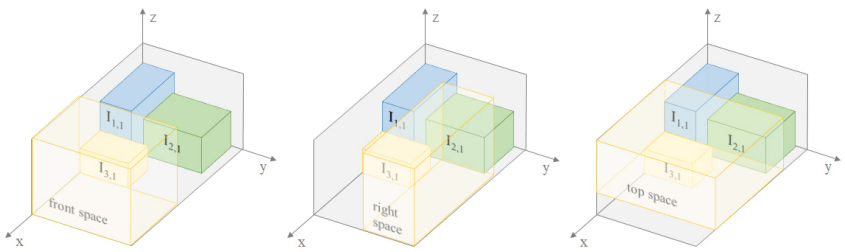
1. fragility flag $f_{i,k}$ (non-fragile first)
2. volume (larger volume first)
3. length $l_{i,k}$ (longer first)
4. width $w_{i,k}$ (wider first).

Algorithm III.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
```

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].



(a) Front Space (b) Right Space (c) Top Space

Figure III.11: New Spaces based on $I_{3,1}$

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. III.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.
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 Section 4) and for the routing heuristic (see Section 5) are listed in Table III.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 III.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 $\mathrm{kg/dm^3}$. 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

Table III.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

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, S. M. W., and Bischoff, E. E. (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, 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.

Table III.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

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.

Table III.8: Comparison best and average results for P1, our Instances

		best			average			Difference		
		sum v_{used}	sum ttd	avg. $time$	sum v_{used}	sum ttd	avg. $time$	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%**

In Table III.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 III.9 presents the best results per instance set.

Table III.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	3,258.14	20.24%	-2.16%

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.
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 III.10 shows the loading constraints as considered in each set. Details are as follows:

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.

Table III.10: Overview of Constraints Sets

Constraints		**Constraints Sets P**									
		basic	Replacing					Adding			Com.
		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

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 III.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 Multiple Overhanging, the number of used vehicles rise by 10.80%, the total travel distance by 8.27%, on average. Since the Top Overhanging 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, 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 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 percent 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**

Table III.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* [s]	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* [s]	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* [s]	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* [s]	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* [s]	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* [s]	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* [s]	-42.53%	-24.94%	-7.83%	-31.01%	-14.11%	-23.71%	-20.65%	-18.92%	**-20.83%**
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* [s]	-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* [s]	54.23%	77.67%	54.45%	101.39%	37.80%	99.29%	50.35%	49.10%	**63.08%**

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 percent. 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. 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 is 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 neighborhood operators on the results as well as impact of instance features on the packing algorithm.

Conflict of interest

The authors declare that they have no conflict of interest.

References

Baker, B., Coffman, E., and Rivest, R. (1980). "Orthogonal Packings in Two Dimensions". In: *SIAM Journal on Computing* vol. 9, no. 4, pp. 846–855. DOI: 10.1137/0209064.

Bischoff, E. E. (2003). "Dealing with Load Bearing Strength Considerations in Container Loading Problems". In.

Bischoff, E. E. and Ratcliff, M. S. W. (1995). "Issues in the development of approaches to container loading". In: *Omega* vol. 23, no. 4, pp. 377–390. ISSN: 03050483. DOI: 10.1016/0305-0483(95)00015-G.

Bortfeldt, A. (2012). "A hybrid algorithm for the capacitated vehicle routing problem with three-dimensional loading constraints". In: *Computers and Operations Research* vol. 39, no. 9, pp. 2248–2257. ISSN: 03050548. DOI: 10.1016/j.cor.2011.11.008. URL: http://dx.doi.org/10.1016/j.cor.2011.11.008.

Ceschia, S., Schaerf, A., and Stützle, T. (2013). "Local search techniques for a routing-packing problem". In: *Computers and Industrial Engineering* vol. 66, no. 4, pp. 1138–1149. ISSN: 0360-8352. DOI: https://doi.org/10.1016/j.cie.2013.07.025. URL: http://www.sciencedirect.com/science/article/pii/S0360835213002404.

Clarke, G. and Wright, J. W. (1964). "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points". In: *Operations Research* vol. 12, no. 4, pp. 568–581. ISSN: 0030364X, 15265463. DOI: 10.1287/opre.12.4.568. URL: http://www.jstor.org/stable/167703.

Fuellerer, G., Doerner, K. F., Hartl, R. F., and Iori, M. (2009). "Ant colony optimization for the two-dimensional loading vehicle routing problem". In: *Computers and Operations Research* vol. 36, no. 3, pp. 655–673. ISSN: 03050548. DOI: 10.1016/j.cor.2007.10.021.

— (2010). "Metaheuristics for vehicle routing problems with three-dimensional loading constraints". In: *European Journal of Operational Research* vol. 201, no. 3, pp. 751–759. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2009.03.046. URL: http://www.sciencedirect.com/science/article/pii/S0377221709002252.

Gendreau, M., Iori, M., Laporte, G., and Martello, S. (2006). "A Tabu Search Algorithm for a Routing and Container Loading Problem". In: *Transportation Science* vol. 40, no. 3, pp. 342–350. ISSN: 0041-1655. DOI: 10.1287/trsc.1050.0145. URL: http://pubsmisc.informs.org/doi/abs/10.1287/trsc.1050.0145.

Junqueira, L., Oliveira, J. F., Carravilla, M. A., and Morabito, R. (2013). "An optimization model for the vehicle routing problem with practical three-dimensional loading constraints". In: *International Transactions in Operational Research* vol. 20, no. 5, pp. 645–666. DOI: https://doi.org/10.1111/j.1475-3995.2012.00872.x. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1475-3995.2012.00872.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1475-3995.2012.00872.x.

Karabulut, K. and İnceoğlu, M. M. (2005). "A Hybrid Genetic Algorithm for Packing in 3D with Deepest Bottom Left with Fill Method". In: *Advances in Information Systems*. Ed. by Yakhno, T. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 441–450. ISBN: 978-3-540-30198-1.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). "Optimization by simulated annealing." In: *Science* vol. 220 4598, pp. 671–80. DOI: 10.1126/science.220.4598.671.

Koch, H., Bortfeldt, A., and Wäscher, G. (Feb. 2018). "A hybrid algorithm for the vehicle routing problem with backhauls, time windows and three-dimensional loading constraints". In: *OR Spectrum* vol. 40. DOI: 10.1007/s00291-018-0506-6.

Krebs, C. and Ehmke, J. F. (2021). "Axle Weights in combined Vehicle Routing and Container Loading Problems". In: *EURO Journal on Transportation and Logistics* vol. 10, p. 100043. ISSN: 2192–4376. DOI: https://doi.org/10.1016/j.ejtl.2021.100043. URL: https://www.sciencedirect.com/science/article/pii/S2192437621000157.

Lodi, A., Martello, S., and Vigo, D. (Nov. 1999). "Heuristic and Metaheuristic Approaches for a Class of Two-Dimensional Bin Packing Problems". In: *INFORMS Journal on Computing* vol. 11, no. 4, pp. 345–357. DOI: 10.1287/ijoc.11.4.345. URL: https://ideas.repec.org/a/inm/orijoc/v11y1999i4p345-357.html.

Mak-Hau, V., Moser, I., and Aleti, A. (2018). "An Exact Algorithm for the Heterogeneous Fleet Vehicle Routing Problem with Time Windows and Three-Dimensional Loading Constraints". In: *Data and Decision Sciences in Action*. Ed. by Sarker, R., Abbass, H. A., Dunstall, S., Kilby, P., Davis, R., and Young, L. Cham: Springer International Publishing, pp. 91–101. ISBN: 978-3-319-55914-8. DOI: 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". In: *Intelligent Decision Support: Current Challenges and Approaches*. Ed. by Bortfeldt, A., Homberger, J., Kopfer, H., Pankratz, G., and Strangmeier, R. Wiesbaden: Gabler, pp. 187–201. ISBN: 978-3-8349-9777-7. DOI: 10.1007/978-3-8349-9777-7. URL: https://doi.org/10.1007/978-3-8349-9777-7.

Moura, A. and Oliveira, J. F. (Oct. 2009). "An integrated approach to the vehicle routing and container loading problems". In: *OR Spectrum* vol. 31, no. 4, pp. 775–800. ISSN: 1436-6304. DOI: 10.1007/s00291-008-0129-4. URL: https://doi.org/10.1007/s00291-008-0129-4.

Pace, S., Turky, A., Moser, I., and Aleti, A. (2015). "Distributing Fibre Boards: A Practical Application of the Heterogeneous Fleet Vehicle Routing Problem with Time Windows and Three-dimensional Loading Constraints". In: *Procedia Computer Science* vol. 51. International Conference On Computational Science, ICCS 2015, pp. 2257–2266. ISSN: 1877-0509. DOI: 10.1016/j.procs.2015.05.382. URL: http://www.sciencedirect.com/science/article/pii/S1877050915011904.

Ratcliff, S. M. W., and Bischoff, E. E. (Feb. 1998). "Allowing for weight considerations in container loading". In: *OR Spectrum* vol. 20, pp. 65–71. DOI: 10.1007/s002910050053.

Reil, S., Bortfeldt, A., and Mönch, L. (2018). "Heuristics for vehicle routing problems with backhauls, time windows, and 3D loading constraints". In: *European Journal of Operational Research* vol. 266, no. 3, pp. 877–894. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2017.10.029. URL: http://www.sciencedirect.com/science/article/pii/S0377221717309426.

Ropke, S. and Pisinger, D. (2006). "A unified heuristic for a large class of Vehicle Routing Problems with Backhauls". In: *European Journal of Operational Research* vol. 171, no. 3. Feature Cluster: Heuristic and Stochastic Methods in Optimization Feature Cluster: New Opportunities for Operations Research, pp. 750–775. ISSN: 0377-2217. DOI: 10.1016/j.ejor.2004.09.004. URL: http://www.sciencedirect.com/science/article/pii/S0377221704005831.

Solomon, M. M. (1987). "Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints". In: *Operations Research* vol. 35, no. 2, pp. 254–265. DOI: 10.1287/opre.35.2.254.

Tarantilis, C. D., Zachariadis, E. E., and Kiranoudis, C. T. (June 2009). "A Hybrid Metaheuristic Algorithm for the Integrated Vehicle Routing and Three-Dimensional Container-Loading Problem". In: *IEEE Transactions on Intelligent Transportation Systems* vol. 10, no. 2, pp. 255–271. ISSN: 1524-9050. DOI: 10.1109/TITS.2009.2020187.

Wei, L., Zhang, Z., and Lim, A. (Nov. 2014). "An Adaptive Variable Neighborhood Search for a Heterogeneous Fleet Vehicle Routing Problem with Three-Dimensional Loading Constraints". In: *IEEE Computational Intelligence Magazine* vol. 9, no. 4, pp. 18–30. ISSN: 1556-603X. DOI: 10.1109/MCI.2014.2350933.

Zhang, D., Cai, S., Ye, F., Si, Y.-W., and Nguyen, T. T. (2017). "A hybrid algorithm for a vehicle routing problem with realistic constraints". In: *Information Sciences* vol. 394-395, pp. 167–182. ISSN: 0020-0255. DOI: 10.1016/j.ins.2017.02.028.

Appendix

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

Instance	**Ceschia et al. (2013) best**		**Our Results 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 III.13: Results for Moura and Oliveira (2009) instances

	Reil et al. (2018) best		**Our Results 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 III.14: Results for Zhang et al. (2017) instances

	Zhang et al. (2017)					Our Results					
	best		average			best			average		
Instance	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
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 III.15: Average Results per Constraint Set, Our Instances

		n			m			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**
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**

Paper IV

Vertical Stability Constraints in Combined Vehicle Routing and 3D Container Loading Problems

Corinna Krebs, Jan Fabian Ehmke

Published in *Computational Logistics*, 12th International Conference, ICCL 2021 Enschede, The Netherlands, September 27–29, 2021 Proceedings DOI: 10.1007/978-3-030-87672-2_29.

Abstract

The vertical stability of the cargo is one of the most important loading constraints, since it ensures parcels from falling on the ground. However, frequently considered constraints either lead to unstable positions, are too restrictive or have high complexity. This paper focuses on the evaluation of different vertical stability constraints, analyses corner cases and introduces a new improved constraint. For the first time, constraints based on the science of statics are considered in the context of combined Capacitated Vehicle Routing Problem with Time Windows and 3D Loading (3L-VRPTW). All constraints are embedded in an established hybrid heuristic approach, where an outer Adaptive Large Neighbourhood Search tackles the routing problem and an inner Deepest-Bottom-Left-Fill algorithm solves the packing problem. For the computational tests, we use a well-known instance set enabling a comparison w.r.t. the number of customers, the number of items and the number of item types. Based on the impact on the objective values and on the performance, we give recommendations for future work.

Contents

1 Introduction

Fueled by the Corona pandemic, the number of shipped parcels worldwide has increased significantly. In 2019, the worldwide parcel volume exceeded the mark of 100 billion parcels for the first time. However, it is estimated to be even between 115 and 132 billion parcels worldwide in 2021. High growth rates are also forecasted for the coming years. Within the next six years, the parcel volume could double (220 to 262 billion) according to PitneyBowes (2020). Along with this, the daily challenges of packing parcels into cargo spaces are intensifying. As a result, the number of conciliation applications to the Bundesnetzagentur (Federal Network Agency for Electricity, Gas, Telecommunications, Post and Railway) in Germany increased by 28% to 1.861 applications in the year 2020 (see BNetzA (2020)). Thereby, 25.3% of applications accounted for damaged consignments. Consequently, the stable loading of parcels is still a major challenge in the planning process and will continue to be an important constraint in the future.

This paper evaluates vertical stability constraints in the field of the 3L-VRPTW, which is a combination of the Vehicle Routing with Time Windows and the 3D Container Loading Problem. In practice, high stability requirements are met by forbidding the parcels to overhang (Full Base Support). Due to this restrictive constraint, other solutions that also achieve stable positions and incur less costs are excluded. For ensuring vertical stability, most approaches require the support of a certain ratio of the base area by directly underlying items. As stated in Ceschia et al. (2013) and in Krebs and Ehmke (2021), this requirement can lead to unstable stacks so that two new vertical stability constraints are regarded. Hence, this paper compares five vertical stability approaches from the literature, which are based on the support of the base area and/or on the science of statics. The latter is evaluated for the first time in the context of the 3L-VRPTW. By indicating common weaknesses and corner cases, we introduce a new vertical stability constraint which enlarges the solution space and is based on the support ratio of the base area and on the science of statics.

We use an established hybrid heuristic for tackling the Vehicle Routing and the Container Loading Problem. The routing heuristic is based on the Adaptive Large Neighbourhood Search (ALNS) by Koch et al. (2018) calling for each route a modified packing heuristic based on the Deepest-Bottom-Left-Fill (DBLF) algorithm proposed by Krebs and Ehmke (2021). For the computational tests, we use a well-known instance set from the literature, where the number of items, item types and customers varies systematically, and evaluate the received results in terms of performance and solution quality in comparison to the established Full Base Support constraint.

The related literature is reviewed in Sec. 2. The considered problem (3L-VRPTW) is formulated in Sec. 3. In Sec. 4, the vertical stability constraints are described in detail. The hybrid heuristic is briefly explained in Sec. 5. Sec. 6 presents computational results analysing the impact of the vertical stability approaches on the 3L-VRPTW. Finally, conclusions are drawn in Sec. 7.

2 Literature Review

In this section, we provide a brief literature overview over different constraints dealing with vertical stability. First, we show the relevant literature in the context of the 3D Container Loading Problem (3D CLP). Then, we present the literature for its combination with the Vehicle Routing Problem (3L-CVRP).

2.1 3D Container Loading Problem

Vertical stability constraints prevent items from falling down on the ground, on top of other items or at the operator while (un-)loading. Various approaches have been formulated in the 3D Container Loading Problem, where a set of items has to be arranged within a container. Most approaches consider the support of the base area, which must be supported either with a defined ratio (Partial Base Support) or completely (Full Base Support) by directly underlying items. This support ratio ranges between 0.55 (Mack et al. (2004)) and 1.0 (Full Base Support, e.g. Ngoi et al. (1994), Fanslau and Bortfeldt (2010), Ceschia and Schaerf (2010)). Since these constraints can lead to either unstable stacks (Partial Base Support) or are too restrictive (Full Base Support), other constraints based on the science of statics have been introduced, where in general the center of gravity of an item or a stack must be supported (e.g. in De Castro Silva et al. (2003), Lin et al. (2006) and Ramos (2015)). Mack et al. (2004) use a combination of the support of the center of gravity and of the item base area w.r.t. items laying at the ground. However, the support of the center of gravity does not guarantee highly stable item stacks as will be demonstrated in Sec. 4.

2.2 3L-CVRP and extensions

When introducing the combined Vehicle Routing and the 3D Container Loading Problem (namely “3L-CVRP"), Gendreau et al. (2006) use the Minimal Supporting Area (aka Partial Base Support) constraint to ensure stable item positions with a support ratio of 0.75. Therefore, this constraint is the most established one in the 3L-CVRP and its extensions. However, Ceschia et al. (2013) show that unstable item stacks can occur when using the Minimal Supporting Area constraint and propose therefore the Multiple Overhanging constraint. In Krebs and Ehmke (2021), several vertical stability constraints, such as the Minimal Supporting Area and the Multiple Overhanging, are examined, and the Top Overhanging constraint is introduced. So far, vertical stability constraints based on the science of statics have not been considered yet in this problem field. Hence, this paper is intended to fill this area.

3 Problem Formulation

Following the formulation as used in Krebs and Ehmke (2021), 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 $h_{i,k}$. Each item has a fragility flag $fg_{i,k}$, grouping items into fragile items $(fg_{i,k} = 1)$ or not fragile items. 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.
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.
It is assumed that each vehicle has a constant speed of 1 distance unit per time unit. If a vehicle arrives at a node 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. The total number of items in a route R_v is described by t_v.
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;

(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 t_v items.

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 the following loading constraints.

(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 t_v of a vehicle does not exceed the maximum load capacity D.

(C5) *LIFO*: No item is placed above or in front of item $I_{i,k}$, which belongs to a customer served after customer i;

(C6) *Vertical Stability*: Each item is placed stable either on the vehicle ground or on top of other items.

(C7) *Fragility*: No non-fragile items are placed on top of fragile items;

The 3L-VRPTW aims 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 constraints.

4 Vertical Stability constraints

In this section, we first explain the calculation of the center of gravity and then, we summarize six constraints for the consideration of vertical stability and present the new Static Stability constraint. Each constraint has been implemented in our solution validator, which can check the feasibility of solutions[1]. The implementation for Ramos (2015) is available here[2].

4.1 Calculation of Center of Gravity

For the approaches based on the science of statics, the center of gravity of an item ($CG_{i,k}$) must be calculated. It is supposed that the mass of an item is homogeneously distributed so that the center of gravity corresponds to the center of volume. Therefore:

$$CG_{i,k} = (x_{i,k} + \frac{l_{i,k}}{2} \quad , \quad y_{i,k} + \frac{w_{i,k}}{2} \quad , \quad z_{i,k} + \frac{h_{i,k}}{2}). \tag{IV.1}$$

Moreover, the center of gravity of a group of items can be calculated. Let G be a set of items belonging to a group. The center of gravity of the group of items is calculated by weighting the center of gravity of each item r ($r \in G$). The equation is as follows:

$$CG_{group} = (\frac{\sum_{r \in G}(x_{CG_r} \cdot m_r)}{\sum_{r \in G}(m_r)} \quad , \quad \frac{\sum_{r \in G}(y_{CG_r} \cdot m_r)}{\sum_{r \in G}(m_r)} \quad , \quad \frac{\sum_{r \in G}(z_{CG_r} \cdot m_r)}{\sum_{r \in G}(m_r)}). \tag{IV.2}$$

[1] see https://github.com/CorinnaKrebs/SolutionValidator
[2] see https://github.com/CorinnaKrebs/StaticStabilityRamos

4.2 Minimal Supporting Area

The Minimal Supporting Area (aka Partial Base Support) constraint is one of the most considered vertical stability constraints in the field of the combined Vehicle Routing and 3D Container Loading Problem (3L-CVRP), since it is included in the original problem formulation by Gendreau et al. (2006). For the Minimal Supporting Area constraint, a certain ratio α of the base area of an item must be supported by the upper surface of the directly underlying items. The parameter α is set to 0.75 in the field of the 3L-CVRP.

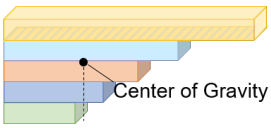


Figure IV.1: Feasible, but unstable stack

However, corner cases can occur leading to theoretically feasible, but actually unstable item arrangements, since it is assumed that all items have the same density. From one level to another, the item's length (or width) is enlarged by $\frac{1}{\alpha}$. This may lead to an item stack as visualized in Fig. IV.1. Since the support is calculated based on the directly underlying items, the stack is feasible. However, when calculating the centre of gravity of the stack (see Eq. IV.2), the dimensions lay outside of the first item. Therefore, the stack would topple.

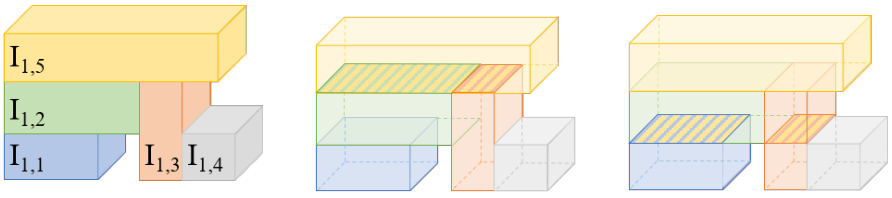
4.3 Full Base Support

As the name suggests, the base area of each item must be fully supported in this constraint. Consequently, overhanging items are not allowed. The approach is the same as for the Minimal Supporting Area though the support parameter α must be one ($\alpha = 1.0$). Apparently, this constraint is the most restrictive one.

4.4 Multiple Overhanging

This constraint has been introduced in Ceschia et al. (2013) and is described in more detail in Krebs and Ehmke (2021). Hereby, all items of a stack are allowed to overhang. However, the Minimal Supporting Area must be obeyed at each level of the stack.

Let $I_{i,k}$ be an item for which the constraint should be checked. First, it is necessary to determine all items which directly or indirectly support item $I_{i,k}$ and store them in set U. An item I_u supports $I_{i,k}$ directly if the upper surface 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 $I_{i,k}$. Based on the set U, the levels are created. Each upper surface of an item I_a ($I_a \in U$) defines a level (see Fig. IV.2b). Another item I_b ($I_b \in U, I_a \neq I_b$) counts to the level if the upper surface of I_b is at the same level as of the level or if the upper surface of I_b is above the level and the base area of I_b is below the level (see Fig. IV.2c). The supporting area units of each level are summed to calculate the total support for item $I_{i,k}$ for each level. Then, it can be checked if each level obeys the Minimal Supporting Area constraint.



(a) Situation (b) First Level (c) Second Level

Figure IV.2: Creation of Levels for Support Area Calculation, exemplary for $I_{1,5}$

4.5 Top Overhanging

This constraint proposed in Krebs and Ehmke (2021) combines the Full Base Support and the Minimal Supporting Area constraint: For all items except the topmost item of a stack, the Full Base Support constraint is applied, while for the topmost item, the Minimal Supporting Area constraint must be obeyed. Consequently, only the topmost item of a stack is allowed to overhang.
The implementation is rather simple: when placing an item $I_{i,k}$ on top of another one, the distance between the upper surface of the last placed item and the vehicle ceiling is calculated. Then, the item with the smallest height of all items not packed yet is determined. If the distance to the ceiling is smaller than the smallest height, it is not possible to pack another item on top of that stack. Therefore, the item $I_{i,k}$ is allowed to overhang obeying the Minimal Supporting Area constraint. Otherwise, another item could be stacked on top of item $I_{i,k}$ and therefore, $I_{i,k}$ must fulfil the Full Base Support constraint.

4.6 Static Stability by Mack et al. (2004)

In the Static Stability constraint introduced by Mack et al. (2004), the centre of gravity of each item is calculated according to Eq. IV.1 and must be supported by the directly underlying items. Moreover, it is required that a certain ratio of the base area of an item is supported indirectly by the items laying on the ground.

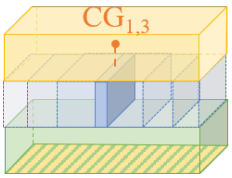


Figure IV.3: Example for a feasible stack applying Mack et al. (2004)

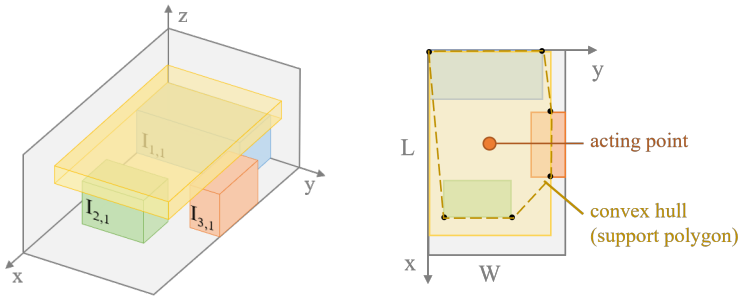
Fig. IV.3 indicates that the item supporting the centre of gravity could be infinitesimally small. This leads to theoretically feasible, but rather unstable stacks. The extreme case where the centre of gravity is only supported by an edge is called the “unstable equilibrium" in the science of statics.

4.7 Static Stability by Ramos (2015)

Ramos (2015) introduces a Static Stability constraint based on the science of statics. In contrast to the Static Stability constraint by Mack et al. (2004), the mass of all items placed above the current placed item $I_{i,k}$ is considered, which can shift the centre of gravity. This new acting point can be calculated as shown in Eq. IV.2.
Then, two cases can occur: If the acting point is supported by directly underlying items, then the position of item $I_{i,k}$ is stable. If the acting point is not supported, then all supporting edges between the item $I_{i,k}$ and its directly underlying items are determined and stored in set T. Based on set T, a convex hull representing the support polygon is calculated. This enables a point-in-polygon to check if the application point lies within the support polygon indicating a stable position for item $I_{i,k}$. If item $I_{i,k}$ is stable according to one of the two cases, the check of the Static Stability constraint is recursively repeated starting from all items directly supporting item $I_{i,k}$ to the items placed on the ground.
Similarly to the constraint by Mack et al. (2004), relatively small items lead to rather unstable stacks.

4.8 New Static Stability

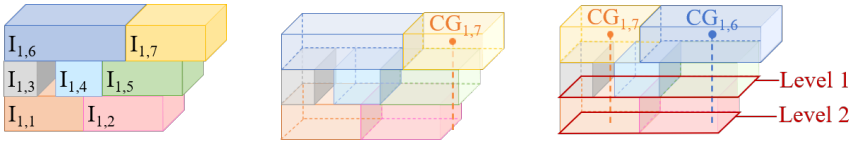
In this paper, we want to introduce a new Static Stability constraint. It is inspired by encountered corner cases in the Minimal Supporting Area constraint, science of statics based approaches, and by the Multiple Overhanging constraint. The main idea is that the centre of gravity of an item must be supported at each level of the stack. A level is created in the same way as described in Sec. 4.4. For each level, the minimum and maximum edges of the level are determined. In particular, the



(a) Situation (b) Convex Hull

Figure IV.4: Example for the Convex Hull Determination

rightest edge, the leftest edge, the foremost edge and the rearmost edge of all items belonging to the level are searched. Then, it is checked whether the centre of gravity is within the frame spanned by the edges (see Fig. IV.5). Therefore, it is not required that the centre of gravity is directly supported as shown in Fig. IV.5c, Level 1. Due to the support of the centre of gravity at each level, item stacks as in Fig. IV.1 are prevented.



(a) Situation (b) Infeasible (c) Feasible

Figure IV.5: Example for Consideration of Static Stability

As shown before, the sole support of the centre of gravity does not ensure highly stable stacks. Therefore, the Minimal Supporting Area constraint with $\alpha = 0.75$ must also be obeyed. This prevents arrangements as shown in Fig. IV.3.

4.9 Summary

The following Table IV.1 gives an overview over the worst-case complexity of the constraints, the role of the support area and the use of the science of statics. Note that t_v corresponds to the number of items in route R_v. As the table indicates, the complexity of the constraints ranges between $O({t_v}^2)$ and $O({t_v}^3)$. However, the restrictiveness and thus the impact on the objective values for each constraint must be determined by means of computational experiments.

Table IV.1: Comparison of Vertical Stability Constraints

	Worst Case Complexity	Consideration of		Corner Cases
		Support Area	Science of statics	
Minimal Supporting Area	$O({t_v}^2)$	✓		✓
Full Base Support	$O({t_v}^2)$	✓		
Multiple Overhanging	$O({t_v}^3)$	✓		
Top Overhanging	$O({t_v}^2)$	✓		
Static Stability by Ramos (2015)	$O({t_v}^2 \log(t_v))$		✓	✓
Static Stability by Mack et al. (2004)	$O({t_v}^2)$	✓	✓	✓
New Static Stability	$O({t_v}^2)$	✓	✓	

t_v = total number of items in R_v

5 Hybrid Algorithm

We implement the previously described vertical stability constraints in a hybrid heuristic approach, which is described in detail in Krebs and Ehmke (2021) and in Koch et al. (2018). In the following, we briefly describe the algorithm and focus on the extensions required.
Since the 3L-VRPTW is a combination of the Capacitated Vehicle Routing Problem with Time Windows (VRPTW) and the 3D Container Loading Problem, the problem is decomposed and separate algorithms are used to solve each subproblem. The routing part is tackled with an Adaptive Large Neighbourhood Search as proposed in Krebs and Ehmke (2021) and in Koch et al. (2018), which is a modification of the heuristic by Ropke and Pisinger (2006a) and by Ropke and Pisinger (2006b). The algorithm is shown in Alg. IV.1.

Algorithm IV.1 Hybrid Heuristic Algorithm

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

construct s_{init} by Savings Heuristic
$s_{best} := s_{curr} := s_{init}$
do
    select number of customers to be removed n_{rem}
    select destroy operator *dest* and repair operator *rep*
    determine next feasible[3] solution $s_{next} := rep(dest(s_{curr}, n_{rem}))$
    for each route R_v **in** s_{next} **do**
        feasible := true
        if Deepest-Bottom-Left-Fill(R_v) not feasible **then**
            feasible := false
            break
        end if
    end for
    if feasible AND Simulated Annealing (s_{next}) **then**
        $s_{curr} := s_{next}$
        update s_{best}
    end if
    update selection probabilities of operators after defined number of iterations
while no Stopping Criterion reached
return s_{best}

The initial solution is constructed by using the Savings Heuristic proposed by Clarke and Wright (1964). In every iteration, it is tried to improve the current solution. Hereby, a set of v_{used} routes is created. The set of routes is feasible if the routing constraints as described in Sec. 3 are obeyed. Each feasible set of routes is evaluated concerning the objective values, whereby the minimization of the number of used vehicles has the highest priority, the total travel distance the second highest priority. A Simulated Annealing approach as proposed by Kirkpatrick et al. (1983) enables the acceptance of inferior feasible solutions in order to enlarge the search.

For each feasible route, the packing algorithm (Deepest-Bottom-Left-Fill algorithm) is called, which is shown in Alg. IV.2. As the name suggests, the items are placed in the deepest, bottommost, leftmost position. Hereby, a list of possible placement spaces is sorted according to the DBL policy. Then, for every item of the route, it is tried to find a feasible position by checking every space of the list until a feasible position is found. A position is feasible if all loading constraints are obeyed. If no feasible position can be found, the Adaptive Large Neighbourhood Search must find a new set of routes in the next iteration. If feasible positions for every item of the set of routes can be found, the feasible solution is stored. The entire algorithm stops either when reaching a defined run time limit, after conducting a defined number of total iterations, or after a total number of iterations without improvement (Stopping Criteria). In the end, the overall best solution is presented.

[3] according to Routing Constraints, see Sec. 3

Algorithm IV.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 AND placement is feasible then
                save placement for I_c
                create new spaces
                sort spaces based on DBL
                erase space sp and too small spaces
                continue with next item
            end if
        end for
    end for
    if no feasible position found then return false
end for
return true
```

6 Computational Studies

In this section, we evaluate the impact of the vertical stability constraints and show their performance in comparison. Hereby, we use our instance set[4] enabling the evaluation w.r.t. the number of customers, items and item types. The instances have either 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). Each instance is tested five times and we present the average results. All results along with detailed packing plans are available via Github[5].
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. In terms of the routing parameters, the same are used as described in Koch et al. (2018). The loading parameters are set according to the values used in the literature.

As shown before, the Full Base Support constraint is the most restrictive one but ensures stable arrangements. The target of this paper is to find a vertical stability constraint that guarantees highly stable positions of items and is less restrictive. Therefore, we exclude approaches where the unstable equilibrium can occur. Consequently, the impact of the constraints Full Base Support, Multiple Overhanging, Top Overhanging and the new Static Stability is investigated. The following Table IV.2 shows the impact of these constraints w.r.t. to the average number of used vehicles (v_{used}), total travel distance ttd and run time in comparison to the Full Base Support constraint. Nevertheless, we tested all approaches described in in Sec. 4 and provide the results online.

As expected, the enlargement of the solution space leads to better objective values (lower number of used vehicles and the total travel distance) for all constraints. Regarding the total level of restrictiveness (impact on the objective values), the following descending order occurs: Full Base Support, Top Overhanging, Multiple Overhanging and Static Stability. Hereby, the Static Stability constraint creates route plans with 8% fewer vehicles and a shorter total travel distance by 5.5% on average. However, compared to the Full Base Support constraint, the Static Stability constraint causes an increase of the run time by 18.56% on average.

Regarding the number of customers or items, the correlations between these instance features and the impact on the objective values are not evident. However, the Static Stability constraint

[4] see https://doi.org/10.24352/UB.OVGU-2020-139
[5] see https://github.com/CorinnaKrebs/Results

Table IV.2: Average Results per Vertical Stability constraint

		n			*m*		*item types*			**Total**
		20	60	100	200	400	3	10	100	
Full Base Support	sum v_{used}	549.20	4,392.40	4,713.60	3,449.40	6,205.80	2,677.40	3,186.40	3,791.40	**9,655.20**
	sum *ttd*	54,956.51	340,558.21	406,434.31	326,656.33	475,292.70	233,987.06	267,122.94	300,839.03	**801,949.03**
	avg. *time* [s]	1,809.61	1,435.49	2,559.26	1,658.60	2,261.04	1,680.67	2,103.83	2,094.97	**1,959.82**
Multiple Overhanging	diff. v_{used}	-6.34%	-11.86%	-1.95%	0.61%	-10.78%	1.53%	-4.80%	-14.13%	**-6.71%**
	diff. *ttd*	-2.28%	-6.90%	-1.71%	0.55%	-7.05%	1.99%	-3.31%	-9.14%	**-3.95%**
	diff. *time*	59.51%	147.96%	40.67%	101.79%	56.35%	102.47%	61.14%	68.51%	**75.58%**
Top Overhanging	diff. v_{used}	-0.95%	-8.36%	1.35%	3.50%	-6.93%	2.14%	-2.24%	-7.78%	**-3.20%**
	diff. *ttd*	0.00%	-4.49%	0.62%	2.29%	-4.26%	2.30%	-1.25%	-4.92%	**-1.59%**
	diff. *time*	61.94%	147.84%	40.55%	102.74%	56.27%	102.64%	61.08%	69.43%	**75.93%**
Static Stability	diff. v_{used}	-5.79%	-9.05%	-7.29%	-4.91%	-9.73%	-0.63%	-6.20%	-14.73%	**-8.01%**
	diff. *ttd*	-2.07%	-6.31%	-5.31%	-3.60%	-6.83%	-0.48%	-4.65%	-10.20%	**-5.51%**
	diff. *time*	24.81%	25.06%	12.71%	24.08%	14.52%	31.94%	14.74%	11.67%	**18.56%**

shows significantly smaller fluctuations than the Multiple Overhanging and the Top Overhanging constraint.

Concerning the number of item types, the Top Overhanging and the Multiple Overhanging constraints lead to an increase in the number of used vehicles and total travel distance by around 2% for instances with three item types. This is due to the fact that the constraints enable overhanging and therefore prevent homogeneous item stacks at the same time. Therefore, gaps between items can occur. Consequently, more vehicles are needed, which also results in a longer total travel distance. As the Static Stability constraint is less restrictive than the Top Overhanging and the Multiple Overhanging constraint, gaps are more likely to be filled. However, as the number of item types increases, the reduction of the objective values is achieved by all constraints. In terms of the Multiple Overhanging constraint, 14.13% fewer vehicles are used, the Top Overhanging shows a decline of 7.78%, the Static Stability of even 14.73%.

All constraints have in common that the average run time increases significantly compared to the Full Base Support constraint. On average, the Multiple Overhanging and the Top Overhanging constraints lead to an increase of the run time of around 75%; for the Static Stability constraint, the increase is only 18.5%. In general, the higher the number of customers or items, the higher the run time. However, according to the results, this is not the case (see $n = 100$ or $m = 400$). The reason is that at the same time, the difference to the maximum run time gets smaller or is exploited.

Based on the described effects, we generally recommend using the Static Stability constraint. However, for time critical or highest stability requirements, the Full Base Support constraint should be used.

7 Conclusion

In this paper, we compared six vertical stability constraints and introduced a new one (Static Stability) in the context of combined Vehicle Routing Problem with Time Windows and 3D Container Loading (3L-VRPTW). The constraints from the literature are based on a defined support ratio of the base area and/or on the support of the center of gravity of each item. We showed that most approaches can lead to unstable item stacks for specific corner cases. Therefore, we introduced a new approach in this paper, which covers common corner cases. This constraint is based on the science of statics and on the support ratio of the base area of an item. As the computational experiments show, the new Static Stability constraint is less restrictive than most of the other approaches and therefore achieves a reduction of the number of used vehicles by 8% and the total travel distance by 5.5% on average, compared to the most restrictive constraint – the Full Base Support. However, the run time increases by almost 19%. Therefore, we recommend the new Static Stability constraint if the reduction of the objective values is of first priority. If the run time or a high stability of items is more important, the Full Base Support constraint should be used.

References

BNetzA (Feb. 2020). *Tätigkeitsbericht Schlichtungsstelle Post 2020*. Tech. rep. Tulpenfeld 4, 53113 Bonn: Bundesnetzagentur für Elektrizität, Gas, Telekommunikation, Post und Eisenbahnen.

Ceschia, S. and Schaerf, A. (Jan. 2010). "Local Search for a Multi-Drop Multi-Container Loading Problem". In: *Journal of Heuristics* vol. 19. DOI: 10.1007/s10732-011-9162-6.

Ceschia, S., Schaerf, A., and Stützle, T. (2013). "Local search techniques for a routing-packing problem". In: *Computers and Industrial Engineering* vol. 66, no. 4, pp. 1138–1149. ISSN: 0360-8352. DOI: https://doi.org/10.1016/j.cie.2013.07.025. URL: http://www.sciencedirect.com/science/article/pii/S0360835213002404.

Clarke, G. and Wright, J. W. (1964). "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points". In: *Operations Research* vol. 12, no. 4, pp. 568–581. ISSN: 0030364X, 15265463. DOI: 10.1287/opre.12.4.568. URL: http://www.jstor.org/stable/167703.

De Castro Silva, J. L., Soma, N. Y., and Maculan, N. (2003). "A greedy search for the three-dimensional bin packing problem: the packing static stability case". In: *International Transactions in Operational Research* vol. 10, no. 2, pp. 141–153. DOI: 10.1111/1475-3995.00400.

Fanslau, T. and Bortfeldt, A. (May 2010). "A Tree Search Algorithm for Solving the Container Loading Problem". In: *INFORMS Journal on Computing* vol. 22, pp. 222–235. DOI: 10.1287/ijoc.1090.0338.

Gendreau, M., Iori, M., Laporte, G., and Martello, S. (2006). "A Tabu Search Algorithm for a Routing and Container Loading Problem". In: *Transportation Science* vol. 40, no. 3, pp. 342–350. ISSN: 0041-1655. DOI: 10.1287/trsc.1050.0145. URL: http://pubsmisc.informs.org/doi/abs/10.1287/trsc.1050.0145.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). "Optimization by simulated annealing." In: *Science* vol. 220 4598, pp. 671–80. DOI: 10.1126/science.220.4598.671.

Koch, H., Bortfeldt, A., and Wäscher, G. (Feb. 2018). "A hybrid algorithm for the vehicle routing problem with backhauls, time windows and three-dimensional loading constraints". In: *OR Spectrum* vol. 40. DOI: 10.1007/s00291-018-0506-6.

Krebs, C. and Ehmke, J. F. (2021). "Axle Weights in combined Vehicle Routing and Container Loading Problems". In: *EURO Journal on Transportation and Logistics* vol. 10, p. 100043. ISSN: 2192–4376. DOI: https://doi.org/10.1016/j.ejtl.2021.100043. URL: https://www.sciencedirect.com/science/article/pii/S2192437621000157.

Lin, J., Chang, C., and Yang, J. (2006). "A Study of Optimal System for Multiple-Constraint Multiple-Container Packing Problems". In: *Advances in Applied Artificial Intelligence*. Ed. by Ali, M. and Dapoigny, R. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 1200–1210. DOI: 10.1007/11779568_127.

Mack, D., Bortfeldt, A., and Gehring, H. (2004). "A parallel hybrid local search algorithm for the container loading problem". In: *International Transactions in Operational Research* vol. 11, no. 5, pp. 511–533. DOI: 10.1111/j.1475-3995.2004.00474.x.

Ngoi, B. K. A., Tay, M. L., and Chua, E. S. (1994). "Applying spatial representation techniques to the container packing problem". In: *International Journal of Production Research* vol. 32, no. 1, pp. 111–123. DOI: 10.1080/00207549408956919.

PitneyBowes (Oct. 2020). "Pitney Bowes Parcel Shipping Index reports continued growth as global parcel volume exceeds 100 billion for first time ever". In: *Pitney Bowes Parcel Shipping Index*. URL: https://www.pitneybowes.com/au/newsroom/press-releases/pitney-bowes-parcel-shipping-index-reports-continued-growth-as-global-parcel.html.

Ramos, A. G. (2015). "Analysis of cargo stability in container transportation". In.

Ropke, S. and Pisinger, D. (2006a). "A unified heuristic for a large class of Vehicle Routing Problems with Backhauls". In: *European Journal of Operational Research* vol. 171, no. 3. Feature Cluster: Heuristic and Stochastic Methods in Optimization Feature Cluster: New Opportunities for Operations Research, pp. 750–775. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2004.09.004. URL: http://www.sciencedirect.com/science/article/pii/S0377221704005831.

— (2006b). "An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows". In: *Transportation Science* vol. 40, no. 4, pp. 455–472. DOI: 10.1287/trsc.1050.0135. URL: https://doi.org/10.1287/trsc.1050.0135.

Paper V

Manual Unloading in 3D Loading Vehicle Routing Problems

Corinna Krebs

Submitted to *EURO Journal on Transportation and Logistics* on 19.07.2023

Abstract

Unloading an item from a delivery vehicle costs time. When modeling loading problems, this time is usually not considered. Rather, feasible unloading is ensured by a simple Last-In-First-Out (LIFO) constraint. The LIFO constraint states that an item is not allowed to be placed on top or in front of items belonging to customers served later. Thus, an item can be unloaded by movements parallel to the vehicle's length.
Following the complexity of real-world applications, this paper relaxes the LIFO constraint by allowing positions of items that block an item from being unloaded. The blocking items must be unloaded and reloaded, causing additional temporal effort. For the first time, this reloading effort is calculated in detail and included in the objective function. In particular, the Methods-Time-Measurement (MTM) approach, a common method for evaluating manual processes in the industrial environment, is employed within 3D Loading Vehicle Routing Problems with Time Windows (3L-VRPTW).
For the computational experiments, the approach is tested in the context of the 3L-VRPTW, which combines the vehicle routing problem with time windows and the 3D container loading. A hybrid heuristic approach is used where an outer Adaptive Large Neighborhood Search tackles the routing problem, and an inner Extreme-Point-Based algorithm solves the packing problem. As the items must have certain properties for enabling manual unloading, a new set of 120 instances is created and used to evaluate the MTM-based approach regarding solution quality and performance.

Contents

1 Introduction

Whenever an item is loaded into or unloaded from a delivery vehicle, the correct item must be identified, lifted, carried, and put down. These actions cost time and depend on the item properties (e.g., dimensions, masses) and the position within the loading space. However, in the literature for container loading and vehicle routing (VRP), this effort was either not been considered (e.g., container loading) or it has been included just as a fixed time component (e.g., service times or handling times in the VRP). Moreover, the unloading process has been simplified so far by applying the Last-In-First-Out (LIFO) constraint. As a result, it is assumed that an item cannot be placed on top or in front of items belonging to customers served later. The intention is that straight movements toward the door accomplish the unloading of items without any obstacles. Hence, the LIFO constraint guarantees a smooth unloading process without any rearrangements of items. However, as a study by Krebs et al. (2023) shows, the LIFO constraint significantly influences the objective values and performance. Therefore, if it was allowed to reload items that block other items during unloading, the solution space could become larger, leading to better objective values and more relevant solutions for logistics practice. In this case, the items which block an item that needs to be unloaded must be unloaded and reloaded. This process costs additional time.
This paper presents an approach for calculating the effort to manually unload items from delivery vehicles. This approach can also be used to relax the LIFO constraint. When items that block others are allowed to be unloaded and reloaded to get access to the desired item, the additional reloading effort can be calculated and included in the objective function. The Methods-Time-Measurement (MTM) is applied to calculate the manual effort. MTM is an approach for analyzing work processes and determining target times (see MTM Association e.V. (2020)). It is used mainly in the industrial environment to plan repeated manual work processes and their estimated duration. MTM can be applied before the manual work is implemented, which supports defining and planning future processes. There are different MTM systems for different industrial applications. One is the MTM Universal Analysis System (MTM-UAS), the most used globally and intended for series production. MTM-UAS is also applied in this paper. It consists of several blocks which combine elementary operations of other MTM systems (e.g., MTM-1). Time values are assigned to the blocks, which depend on various time-influencing factors. These blocks can be summed up to represent the total time-effort of a process, such as unloading and reloading items from and into a delivery vehicle.
This paper aims to investigate whether relaxing the LIFO constraint by allowing reloading of blocking items is worthwhile compared to the additional efforts as indicated by the MTM system. This paper contributes in four aspects:

1. For the first time, the MTM approach is customized for the individual calculation of the manual unloading effort of items in delivery vehicles;

2. The LIFO constraint is treated as a soft constraint so that reloading of items is allowed;

3. The effort of unloading and reloading items is calculated and becomes part of the objective function to optimize solutions in this regard;

4. A new instance set respecting manual unloading is created, and the impact of the unloading effort is evaluated in computational experiments.

For the computational tests, the MTM approach is included in the combined 3D loading and vehicle routing problem with time windows (3L-VRPTW), solved through a hybrid algorithm. Specifically, the approach by Krebs et al. (2023) using a hybrid algorithm approach is implemented, containing the Adaptive-Large-Neighborhood Search for solving the routing problem and the Deepest-Bottom-Left-Fill algorithm as a packing algorithm. The objective function is thereby adapted to consider the total travel time with unloading efforts. Furthermore, the packing algorithm is adapted using efficient aspects of the packing algorithm by Gajda et al. (2022), which considered the number of blocking items in the objective function.
The paper is organized as follows: In Section 2, the relevant literature is reviewed. The 3L-VRPTW is formulated in Section 3. In Section 4, the MTM approach and the corresponding calculations are presented. Section 5 describes a hybrid solution algorithm, and Section 6 contains the computational experiments. Finally, Section 7 outlines the paper's conclusions.

2 Literature Review

This section presents the literature relevant to this paper. First, an overview of investigated algorithms and constraints in the context of the 3L-VRPTW is given. This is followed by relevant literature concerning the constraints that deal with the unloading process and which offer avenues for additional research.

2.1 3L-VRPTW

In this paper, the unloading and loading of vehicles are analyzed in the context of the 3L-VRPTW, as its objective function can be adapted to consider the unloading effort. The 3L-VRPTW is an extension of the 3L-CVRP. The latter variant without time windows was introduced by Gendreau et al. (2006). The same loading constraints are considered in both optimization problems: items are orthogonally packed into the vehicle loading space (Orthogonality constraint) considering non-overlapping and their dimensions (Geometry). Items can be rotated only along the width-length plane (Rotation constraint). Each item has a mass, and the vehicle has a maximum capacity (Load Capacity). Items are grouped into fragile and non-fragile items to prevent stacking fragile items on top of each other (Fragility constraint). Each item must have a specific support percentage according to its base area (Minimal Supporting Area constraint). When unloading items, this should be done by direct movements parallel to the length of the vehicle without rearranging other items (LIFO constraint).

Moura (2008) and Moura and Oliveira (2009) introduce the Vehicle Routing with Time Windows and Loading Problem (VRTWLP). This problem represents the 3L-VRPTW without the consideration of item masses and the fragility of items, the requirement of higher stability requirements (full support), and more rotation options. A Multi-Objective Genetic Algorithm is suggested by Moura (2008) to generate routes. In addition, a wall-building heuristic is invoked to address the loading problem whenever a customer is added to a route. In Moura and Oliveira (2009), which combines a hierarchical and sequential method, this packing heuristic is also applied. While the sequential method tackles the VRPTW and bin packing, the hierarchical one primarily solves the VRP. Finally, an instance set of 46 instances is introduced and tested for computational experiments.

In Reil et al. (2018), a Tabu Search algorithm addresses the packing problem. A Multi-Start Evolutionary Search reduces the number of used vehicles, and a different Tabu Search reduces the total travel distance. This algorithm currently obtains the best-known results for the set of instances from Moura and Oliveira (2009). Zhang et al. (2017) introduce the 3L-VRPTW and also published an instance set. They solve the problem with a hybrid algorithm, which uses a new loading heuristic and a routing heuristic based on a Tabu Search and an Artificial Bee Colony algorithm. Krebs et al. (2021) present new best-known results for the 3L-VRPTW. They focus on the introduction of new complex loading constraints and their evaluation. For the computational tests, a hybrid algorithm is used, consisting of the Adaptive Large Neighborhood Search proposed by Koch et al. (2018) for solving the routing problem and the Deepest-Bottom-Left-Fill algorithm shown in Krebs et al. (2023).

This paper uses the hybrid algorithm proposed by Krebs et al. (2023) with an adapted objective function to include the unloading effort. Moreover, it incorporates more loading constraints (e.g., Axle Weights, Balanced Load, and Reachability) as described in Krebs et al. (2021).

2.2 Unloading Constraints

The LIFO constraint, also known as the Sequential Loading and Multi-Drop constraint (Bischoff and Ratcliff 1995), requires that the requested items be available at each customer location without rearranging the others. Thus, items are loaded such that they can be unloaded through straight movements parallel to the length of the vehicle. Therefore, no item demanded by a later-visited customer may be placed over or between the item to be unloaded first or near the vehicle's rear (Gendreau et al. 2006).

As the LIFO constraint is rather restrictive, Tarantilis et al. (2009) propose a relaxation by allowing an item to hang over other items served later. The idea is that items that are unloaded manually can be pulled out and must not be lifted by lifting equipment. They introduce the new variant "Capacitated Vehicle Routing Problem with Manual 3D Loading Constraints" (M3L-CVRP). As a hybrid algorithm, they use a combination of Tabu Search and Guided Local Search to build the routes. Six packing heuristics are called successively for the Loading Problem until a feasible solution has been found. The adapted LIFO constraint is also integrated with a paper by Ceschia et al. (2013) and evaluated by Krebs et al. (2021). Although the solution space is larger by allowing item overhanging, the objective values do not improve. Consequently, this adaption of the LIFO constraint is not effective.

Another constraint dealing with the unloading process is the Reachability constraint. In the context of the Three-Dimensional Bin Packing Problem, Junqueira et al. 2013 develop this constraint to ensure items remain reachable during unloading or arranging operations to avoid the driver standing on items. In addition, this constraint is included in Ceschia et al. (2013); its impact is tested in Krebs et al. (2021). The latter evaluation shows that the Reachability constraint increases the total travel distance by around 3% on average. Therefore, in this paper, the Reachability constraint is adapted and included in the model.

Gajda et al. (2022) focus on a variant of the 3D Single Knapsack Problem (3DSKP) where a number of items is available and must be selected and packed into a single container. They consider the loading constraints as in the 3L-VRPTW and consider axle weights of vehicles, the balanced load of the cargo, items with different priorities, as well as dangerous items which must be placed next to an unloading point. In their paper, the LIFO constraint is treated as a soft constraint, allowing rearrangements of obstacles and making the obstacle number relevant. Their multi-objective function consists of three components: 1) to maximize the taxability (taxable weight), defined as the maximum between the total cargo weight and the volume multiplied by a constant; 2) to minimize the number of obstacles; and 3) to maximize the cumulative item priorities. To solve the problem, they propose a randomized constructive heuristic that consists of several iterative phases: pre-processing procedure for combining items, the item-sorting phase, randomization for partially perturbing the sorting, and the conduction of the packing.

However, in contrast to Gajda et al. (2022), where the number of obstacles is included in the objective function, this paper considers the obstacles by calculating the time-wise effort and adding it to the objective function. Instead, the objective is evaluated in the context of a combined 3D loading and routing problem with time windows (3L-VRPTW), as introduced by Zhang et al. (2017). Moreover, in addition to the constraints considered in Zhang et al. (2017), the axle weights and the balanced load as in Krebs et al. (2021), and the Reachability constraint as proposed by Ceschia et al. (2013), are included.

3 Problem Formulation

This section deals with the problem formulation of the 3L-VRPTW, used to demonstrate the impact of the relaxed LIFO constraint. First, for this problem, the unloading and reloading effort is calculated and evaluated. Then, following the convention by Koch et al. (2018), the 3L-VRPTW is described as follows.

Suppose that $G = (N, E)$ is a complete, directed graph, where N is the set of $n + 1$ nodes, including the depot (node 0) and the n customers to be served (nodes 1 to n). Let E be the set of edges linking each pair of nodes. Each edge $e_{i,j} \in E$ $(i \neq j, i, j = 0, ..., n)$ has an associated routing distance $d_{i,j}$ with $d_{i,j} > 0$. The demand of customer $i \in N \setminus \{0\}$ is represented by c_i cuboid items. Let the sum of all the demanded items be m. In addition, three times are assigned to each node i to account for time windows: the ready time RT_i, which is the earliest possible start time; the due date DD_i representing the latest possible start time, and the service time ST_i specifies a required fixed time to start the unloading process for a customer i. A vehicle must wait until the ready time is reached if it arrives at an edge before that time. Each item $I_{i,k}$ $(k = 1, ..., c_i)$ is described by its four properties: length $l_{i,k}$, width $w_{i,k}$, height $h_{i,k}$, mass $m_{i,k}$, and a fragility flag $f_{i,k}$ (fragile: $f_{i,k} = 1$). A v_{max} homogeneous vehicle fleet is available for item delivery. Each vehicle v has a maximum payload D and a cuboid loading area with dimensions of L, W, and H. The speed of each vehicle is one distance unit per time unit. The number of used vehicles in a solution is given by v_{used}. A solution is represented through a set of v_{used} routes R_v $(v = 1, ..., v_{used})$ and packing plans PP_v, where R_v is an ordered sequence of at least one customer, and P_v is a packing plan that specifies where each item is placed inside the loading space.

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;

(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 fulfill the following loading constraints:

(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;

(C5) *Fragility*: No non-fragile items are placed on top of fragile items;

(C6) *LIFO*: No item is placed above or in front of item $I_{i,k}$, which belongs to a customer served after customer i;

(C7) *Minimal Supporting Area*: Each item has a supporting area of at least a percentage α of its base area;

(C8) *Reachability*: The distance between an item and the operator must be less than or equal to 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-VRPTW consists of determining a feasible solution minimizing the total travel time (ttt) and obeying all the above constraints. For the calculation of the total travel time, the traveled time based on the total travel distance, the waiting times in case of arrival before ready time, and the service time are included. Additionally, this paper considers the unloading effort for items, which is also respected in the total travel time. Accordingly, the following section outlines how this unloading effort is modeled.

4 Unloading Effort

This section introduces the MTM approach to calculate the unloading effort. Specifically, the details concerning the unloading processed and the formula for calculating the unloading effort per item are described and discussed, respectively. In addition, the implementation of the relaxed LIFO constraint is presented in the following.

4.1 Methods-Time-Measurement

In the last 20 years, the design of work systems and organizations has continually changed, particularly in reducing the cycle times of lines (primarily in the automobile sector and its suppliers). Per the above changes, modern human labor is characterized by cyclical repetition in numerous tasks. For a realistic evaluation of the work processes, predetermined motion time systems have been developed. For example, the MTM system, developed by the MTM Association (MTMA), introduced several block-building systems. These systems share the conceptualization that several blocks representing movements are linked to time values. These times depend on several factors, e.g., the mass of an object, the movement length, and the position accuracy. Consequently, several time values per object and movement property exist. An associated code is given to identify the selected block and property (MTM code). The time values are in Time Measurement Unit (TMU), where 1 TMU corresponds to 0.0006 min. All data is available on data cards.
The MTM system MTM-1 forms the basis of other MTM building block systems. MTM-1 contains five basic motion elements: Reach, Move, Grasp, Release, and Position. Combining these basic elements makes it possible to describe 85% of all activities involved in the manual assembly context (Rückert et al. 2021). MTM-UAS is based on MTM-1 and was developed by statistically combining data from the MTM-1's basic elements. For example, MTM-UAS has a block named "Get and Place", which consists of MTM-1 basic motions of Reach, Grasp, Move, Position, and Release. This paper uses the MTM-UAS data card to calculate the unloading effort, an approach presented in the next section.

4.2 Calculation of Unloading Effort

This section describes the necessary steps to compute the effort for unloading and reloading an item with MTM. In particular, we assume that the process of unloading and reloading consists of the following five steps:

1. **Visual Control:** The correct item must be identified. Therefore, the operator must check the item visually (e.g., by reading the address label).

2. **Get:** After identifying the correct item, it must be lifted and pulled out. The operator now has control over it.

3. **Movement:** The operator now holds the item. It must be carried to the door of the vehicle loading space. The walking distance corresponds to the distance between the position of the item and the door.

4. **Place:** After reaching the door of the vehicle loading space, the item is put down. An exact position is not necessary; an approximate placement is sufficient.

5. **Movement:** The operator walks back into the vehicle loading space.

After this description of the unloading process, it is necessary to find the equivalent blocks in the MTM-UAS data card.[1]

Table V.1: Extract of MTM-UAS Data Card

Action	Detail	MTM Code	TMU
Visual Control		VA	15
Get and Place	>1 kg to ≤ 8 kg	AH	55
	>8 kg to ≤ 22 kg	AL	115
Body Movement	Walk per m	KA	25

Table V.1 shows an extract of the necessary values from the MTM-UAS data card, matching the described actions. The “Get and Place" MTM block combines the previously described actions #2 and #4. Its time depends on three properties: The item's mass, motion length, and positioning accuracy. Concerning the first aspect, the item's mass is classified into two ranges: between 1 and 8 kg or 8 and 22 kg. There are several ranges for the motion length. Here, the highest class (> 50 to ≤ 80 cm) is always chosen due to the consideration of the item's dimensions and the operator's arm length. The last aspect considers positioning accuracy because exact positioning costs more time. However, for this use case, such a precise positioning is not necessary. Consequently, the unloading effort for an item I_p placed at position (x_p, y_p, z_d) is as follows:

$$\text{effort}_d = \text{Visual Control} + 2 \cdot \text{Walk} \cdot (L - x_p) + \text{Get and Place}(m_d) \tag{V.1}$$

The same equation can be used to calculate the loading effort.

4.3 Implementation of Reloading

The previous section described the calculation of the unloading effort for one item. For every item, the unloading effort must be calculated at least once. However, in the case of the relaxation of the LIFO constraint, blocking items are intended to be unloaded and reloaded. Hence, this section describes the prerequisites for reloading and the selection of items for reloading.

As a short recap, the LIFO constraint's inclusion ensures an unloading process without obstacles in the related literature. The constraint states that an item should not be placed on top or in front of items belonging to customers served later. However, this constraint greatly impacts the objective values (see Krebs et al. (2023)). In this paper, items are allowed to block other items during the unloading process. The idea is that the blocking items are temporarily unloaded to access the item that needs to be unloaded I_p. After unloading the item for delivery I_p, the blocking items are reloaded again.

The first prerequisite for unloading is that the position of I_p is infeasible according to the LIFO constraint. If this is the case, then the applicability of reloading is examined; if not, the examination does not occur. The second prerequisite is that I_p is not placed underneath other items. Otherwise, the items placed on top of I_p would need to fly after unloading I_p as its base is missing, or another position for each of these items must be found, leading to a decrease in performance.

If these two prerequisites are given, the reloading approach is considered. The unloading and reloading effort can be calculated using the previously introduced Eq. V.1. However, it is an open question about which items must be reloaded. This is described in the following. In general, the number of reloading items should be minimized. For example, when an item I_p should be unloaded for delivery, it can be accessed from three sides: From the left side, the front side, and the right

[1] The complete MTM-UAS data card can be downloaded here: https://mtm.org/fileadmin/mtm_upload/Download/MTM-UAS_data_card_EN.pdf

side. Consequently, three straight corridors from the door to I_p are available. This corridor must provide a minimal width c so the operator and item can fit through. Therefore, the operator's width (60 cm) or the item's width w_d (if wider) are utilized to determine the width of the corridor c:

$$c = \max(w_p, 60). \tag{V.2}$$

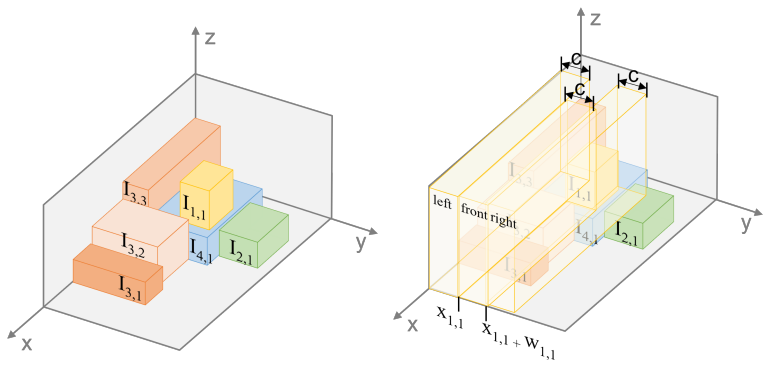
Thus, there are three corridors (left, front, right), which have to obey the minimal width c to access the item I_p for delivery. Table V.2, as follows, shows the corridor dimensions based on the item I_p. All corner points must lay inside the vehicle loading space.

Table V.2: Definition of corridors for accessing item I_p

Corridor	Minimal Corner	Maximal Corner
Left	(0, $y_p - c$, 0)	(L, y_p, H)
Front	(0, y_p, 0)	(L, $y_p + c$, H)
Right	(0, $y_p + w_p$, 0)	(L, $y_p + w_p + c$, H)

Based on these three corridors, the items lying within these can be determined. Here, three aspects must be considered: the point in time of unloading (1), the vertical support of the unloaded item I_p (2), and the reachability (3). Concerning the point in time, all items unloaded before the intended item I_p are irrelevant. The same applies to items already considered for reloading at this point in time. Concerning 2 – vertical support, items that contribute to the support of I_p to more than 50% are also excluded from the reloading option as these items are crucial for the vertical stability of I_p. Lastly, a corridor can be excluded from the investigation if one of its items blocks the item I_p so that it is not reachable, thus violating the Reachability constraint (C8).

Fig. V.1 shows an example of the definition of corridors in this context. The item which must be accessed for delivery is $I_{1,1}$. All items are unloaded later than $I_{1,1}$. The *left corridor* contains three items: $I_{3,1}$, $I_{3,2}$, $I_{3,3}$; the *right corridor* contains only $I_{2,1}$. In addition, the *front corridor* contains $I_{3,1}$ and $I_{3,2}$. The item $I_{4,1}$ is excluded from the corridors, as it fully supports item $I_{1,1}$, thus contributing to more than 50% of the support of item $I_{1,1}$. Consequently, for this situation, the *right corridor* should be selected to access item $I_{1,1}$, as it has the fewest number of items.



(a) $I_{1,1}$ to be unloaded (b) Three Corridors

Figure V.1: Rearrangement Corridors for Item $I_{1,1}$

After selecting the corridor, the effort according to Eq. V.1 is applied to each item within the corridor twice: once for unloading and once for reloading. This procedure gives the total effort for unloading I_p at its delivery time.

5 Hybrid Solution Approach

This section describes the hybrid algorithm for solving the 3L-VRPTW. Since the 3L-VRPTW combines the Capacitated VRP (CVRP) and the 3D Container Loading Problem, the hybrid

algorithm consists of two separate algorithms targeting each subproblem. The algorithm is based on the hybrid heuristic described in Krebs et al. (2023). The calculation of the unloading effort is implemented within the packing heuristic. However, to understand the entire framework, the connection between the routing and the packing algorithm is described roughly in the following and detail in the subsections.

First, a set of routes is created by the routing algorithm. It takes the routing constraints (R1, R2) into account. For each feasible route, the packing algorithm is then called, which tries to create a feasible packing plan considering the loading constraints. In this part, the relaxation of the LIFO constraint also takes place. If no feasible packing plan can be created for a route, a new set of routes must be found.

The following three algorithms are described in detail: the routing heuristic, the packing heuristic, and the loading constraints feasibility check. The suitable line numbers are given in square brackets.

5.1 Routing Heuristic

For solving the VRP, the modified Adaptive Large Neighbourhood Search (ALNS), described in detail in Koch et al. (2018), is used. Initially, the algorithm was developed to solve the VRPTW with Backhauls, and the objective function optimizes the total travel distance. Both aspects are adapted to meet the problem formulation described before. The general framework is shown in Alg. V.1; the corresponding line number of the algorithms is given in square brackets. Further details are described in the original paper.

Algorithm V.1 Adaptive Large Neighbourhood Search

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

1: construct initial solution s_{init}
2: $s_{best} := s_{init}$
3: $s_{curr} := s_{init}$
4: **do**
5: select removal operator rem
6: select insertion operator $inst$
7: select number of customers to be removed n_{rem}
8: determine next solution $s_{next} := inst(rem(s_{curr}, n_{rem}))$
9: check acceptance of s_{next}
10: **if** s_{next} is accepted **then**
11: $s_{curr} := s_{next}$
12: **if** $f(s_{curr}) < f(s_{best})$ **then**
13: $s_{best} := s_{next}$
14: **end if**
15: **end if**
16: **if** $iter_p$ reached **then**
17: update selection probabilities for insertion and removal heuristics
18: **end if**
19: **while** one stopping criterion is not met

5.1.1 Initial Solution

The Savings Heuristic developed by Clarke and Wright (1964) is used to construct the initial solution s_{init} [1]. This first solution complies with all loading and routing constraints (except S3). Then, the ALNS determines additional feasible and better solutions based on this initial feasible set of routes.

5.1.2 Iteration

One removal operator (rem) and one insertion operator ($inst$) are randomly selected for each ALNS iteration [5–6]. By removing and then re-inserting a number of customers from the solution, the

operators are used to construct the next solution s_{next} [8]. The number of customers to be removed n_{rem} ($n_{min} \leq n_{rem} \leq n_{max}$) is picked randomly [7]. The removal and insertion operators are described in the belonging Appendix. The resulting solution is then tested to verify that it complies with the routing constraints [9] listed in Section 3. Moreover, the packing algorithm, described in the next section, is invoked.

5.1.3 Evaluation Function

The evaluation function leads the search by enabling the comparison of feasible and infeasible solutions; it additionally provides the total routing costs. The evaluation is mainly based on the objective function, e.g., minimizing total travel time. As described before, the total travel time includes the total travel distance, waiting times, service times, and unloading effort. Based on this, penalties are added for customers that have yet to be dispatched N_{miss} and for exceeding the number of used vehicles ($v_{max} - v_{used}$). Each customer i which has yet to be dispatched ($i \in N_{miss}$) is assigned to one vehicle (round trip), even if this leads to exceeding the number of used vehicles. The penalty term pen_v is used to achieve a reduction of the used vehicles v_{used}. The evaluation function for a solution s is as follows:

$$\begin{aligned} f(s) &= ttt(s) + pen_v \cdot max(0, v_{used} + |N_{miss}| - v_{max}) \\ &+ \textstyle\sum_{i \in N_{miss}}^{N_{miss}} (max(d_{0,i}, RT_i) + d_{i,0} + ST_i). \end{aligned} \tag{V.3}$$

5.1.4 Solution Acceptance

The smaller a solution's evaluation function value, the better it is rated. A better and more feasible solution is always accepted, while a worse solution may be accepted [9] depending on a Simulated Annealing Heuristic proposed by Kirkpatrick et al. (1983). The acceptance probability is thereby 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]. 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.

5.1.5 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 $iter_{wimpr}$;
- runtime limit t_{max}.

5.2 Packing Heuristic

This section describes the packing heuristic. It combines two algorithms: The main basis is the Deepest-Bottom-Left-Fill algorithm, as proposed by Krebs et al. (2023), supplemented with elements of the Extreme-Point-Based algorithm shown in Gajda et al. (2022). Before describing the packing algorithm, information concerning the vehicle loading space is relevant. In this regard, the origin of a Cartesian coordinate system is positioned in the cargo space's deepest, leftmost corner, with the driver's cab placed behind it. The cargo space is parallel to the x, y, and z axes in length, width, and height. The placement of an item $I_{i,k}$ is defined by $(x_{i,k}, y_{i,k}, z_{i,k})$ of the corner closest to the origin.

The algorithm is shown in Alg. V.2. It is called by the routing heuristic for each route of a solution. The packing algorithm's first step [1] is to create the set of items to be packed IS. The items are sorted observing 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).

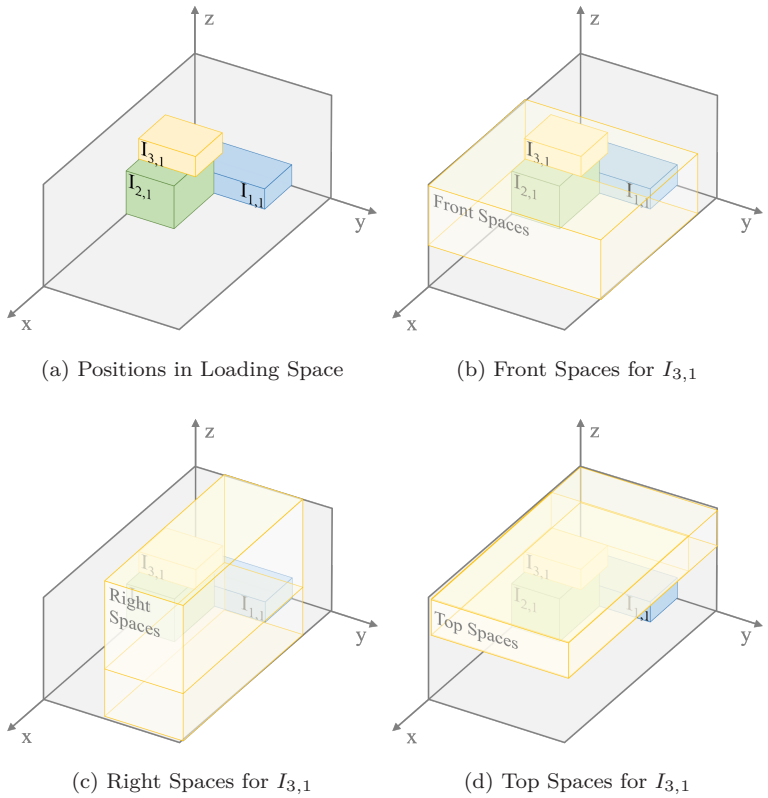
If the reloading approach is not allowed and thus, the LIFO constraint is treated as a hard constraint, the item sequence IS equals demanded items in the reversed customer's visiting order.

Algorithm V.2 Deepest-Bottom-Left-Fill Algorithm with Retry

Input: Instance Data
Output: Feasible placements for items (PP_v)

```
initialize sorted sequence of items IS
initialize set of unique available spaces S
for each item I_p ∈ IS do
    sort spaces
    for each space sp ∈ S do
        if support of space sp for I_p < α then continue
        for each permitted orientation do
            if item I_p fits in space sp then
                if ConstraintsFeasibilityCheck(placement I_p) = true then
                    save placement for I_p
                    create new spaces
                    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 w.r.t. l_min and h_min then
                            erase space si
                        end if
                    end for
                    break
                end if
            end if
        end for
    end for
    if no feasible position found then
        if first trial then
            move I_p to end of S
        else
            return false
        end if
    end if
end for
return true, PP_v
```

Concerning the placement positions, available free spaces in the vehicle loading space are used. Let S be a set containing unique cuboids representing these spaces. Initially, this set consists of one space representing the total cargo space [2]. As a result, the first item in the packing sequence IS is assigned to the origin.
Each space sp of the set is tested as a possible item position [9] w.r.t. to the other loading constraints. This part is presented in more detail in Alg. V.3. If a feasible position is found, the position is stored for the item [10]. Fig. V.2 visualizes the spaces for an item I_p, which are created as follows [11]:
First, the *Front Space* is defined by the item's front edge (minimum x-value) and either the cargo space door or the nearest item in front of the item (maximum x-value). Then, the minimum and maximum values for the y-axis are determined by the cargo space or other items; the minimum and maximum values for the z-axis are searched similarly (see Fig. V.2b). Next, the *Right Space*



(a) Positions in Loading Space

(b) Front Spaces for $I_{3,1}$

(c) Right Spaces for $I_{3,1}$

(d) Top Spaces for $I_{3,1}$

Figure V.2: Space Creation

is bounded along the y-axis by the left side of the item (minimum y-value) and either by the cargo space wall or by the rightmost item (maximum y-value). Based on these limitations, the minimum and maximum values for the x-axis are determined, then for the z-axis (see Fig. V.2c). Finally, the *Top Space* is defined by the item's top surface and either the cargo space ceiling or an item overhanging over the current item I_p. In the next step, the minimum and maximum values for the y-axis and z-axis are determined by the cargo space or other items (see Fig. V.2d). Besides that, three additional spaces are created if they are unique: another Front and Right Space, where the minimum z-value represents the bottom edge of item I_p and another Top Space, where the minimum x-value is the deepest edge of item I_p. These additional spaces are also presented in Fig. V.2 for the Right and Top Spaces; for the Front Space specifically, the additional space corresponds to the previous one and is thus not unique. The definition and determination of the placement positions have two comparable aspects as in the algorithm by Gajda et al. (2022): First, the algorithm by Gajda et al. (2022) is based on extreme points, which means that items are placed next to the extreme points—the DBLF algorithm uses the same principle. Second, when creating the Left and Front Spaces, the spaces are extended along the z-axis as described before. This accords with the projection approach by Gajda et al. (2022).

If a possible position for item I_p has been identified, the corresponding space sp, wherein the item I_p is placed, is removed from the set S [12]. To increase the performance, only spaces whose length and height are large enough for the smallest dimensions of any item among all items of the route IS are included in the set S. Therefore, the shortest length or width l_{min} and height h_{min} of any unplaced item of the route IS are searched [17]. 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 [13-18]. At the same time, all spaces are evaluated regarding their intersection with item I_p. In the case of intersection, the minimum and maximum values for the x-, y-, and z-axis are decreased [15]. This process ensures that when placing an item within a space, this item does not overlap with other items or the vehicle walls (C1). Therefore, an

additional overlapping check for the Geometry constraint (C1) is not required.
If a feasible position for I_p was found, the position for the following item is searched [32]. If all spaces are checked once and no feasible position for item I_p is found [24], the item is moved to the end of the set IS to retry this item once again [25-27]. This retry list is also proposed by Gajda et al. (2022). After all of the items have been considered (i.e., the first loop), those that could not be placed initially are tried. The idea is that more spaces will be available, potentially leading to a feasible position. According to Gajda et al. (2022), who found that the benefit of multiple retries is usually negligible, this single additional retry is sufficient. If no position can be found for an item on the second attempt [28-30], the route is considered unpackable and thus rejected. If this occurs, the ALNS must generate a new set of routes.

5.3 Constraints Feasibility Check

This part deals with the Constraints Feasibility Check, where the position of item I_p is checked w.r.t to the feasibility of the loading constraints as described in Section 3. The following algorithm Alg. V.3 is called in the previously presented packing algorithm Alg. V.2 (see line 9). Four constraints are obeyed indirectly before calling this algorithm:

1. Geometry (C1) prevents items from overlapping with each other or the vehicle wall. This constraint is obeyed by creating spaces where the dimensions are directly adapted.

2. Orthogonality (C2) deals with placing items only orthogonal to the vehicle walls. This is also obeyed through the definition of the spaces that are parallel to the vehicle walls.

3. Rotation (C3) constraint allows the rotation of items only along the width-length plane. This is enabled in Alg. V.2, line 7.

4. Load Capacity (C4) checks that the sum of masses per tour does not exceed the capacity limit D. This is checked during the next solution creation process (Alg. V.1, line 8)), along with all routing constraints.

Consequently, the constraints Fragility, Minimal Supporting Area, LIFO, Reachability, Axle Weights, and Balanced Load must still be checked. These constraint checks are sorted according to the algorithmic complexity, determined by Alg. V.3. Therefore, it starts with the feasibility check for Axle Weights [1] and the Balanced Load constraint [2]. The implementation of both constraints is described in detail in Krebs et al. (2021). After that, the Fragility and the Minimal Supporting Area constraint checks are executed. As the LIFO constraint is relaxed, a deep-dive of this check follows.

First, it is necessary to initialize the corridor sets (Left Corridor LC, Right Corridor RC, Front Corridor FC) [4]; these will contain the items of the corridor [5]. Then, the reachability flags are initialized [6]. These flags indicate whether a corridor can be reached (true) or an item blocks the corridor, making the item I_p is not reachable (false). Consequently, there are three flags—one per corridor. In the next step, the width of the corridor c is determined [7]. The LIFO constraint is checked for the position of item I_p [8]. Then, if the position is feasible regarding LIFO, the item's reachability is checked [17]. The implementation details can also be found in Krebs et al. (2021). If the position is not feasible w.r.t. LIFO and reloading is allowed, then the reloading approach starts. For every already placed item, I_t is checked regarding whether it lays within a corridor [13, 19, 25]. Thus, items unloaded before the item I_p [11] or are already rearranged for when I_p should be unloaded [12] are ignored. After that, three similar blocks [13-18, 19-24, 56-30] follow – one for each corridor (Left, Front, Right). Each block starts with defining the corridors [13, 19, 25], as previously presented in Section 4.3, and checking if an item is inside the corridor.

If an item lays inside a corridor, the support of the item on the item I_p is examined: if the support for item I_p is more than 50%, the item I_t cannot be rearranged as this item is crucial for the vertical stability of I_p, preventing it from falling on to the ground. Thus, as item I_t is within the corridor but cannot be rearranged for stability reasons, it is necessary to determine if item I_p is reachable despite it blocking item I_t. The reachability check depends on the corridor—thus, on the side of item I_p—being accessible: For the front corridor, the distance between the front edges is determined and must be below λ, the parameter representing the maximum distance. Then, the

Algorithm V.3 Constraints Feasibility Check

Input: Instance Data, item I_p
Output: Feasibility placement for item I_p, Unloading Effort

1: **if** Axle Weights Constraint Check(I_p) = **false return false**
2: **if** Balanced Load Constraint Check(I_p) = **false return false**
3: **if** Fragility Constraint Check(I_p) = **false return false**
4: **if** Minimal Support Constraint Check(I_p) = **false return false**
5: LC, RC, FC := ∅ ▷ Initialize Corridor Sets
6: r_{LC}, r_{RC}, r_{FC} := true ▷ Reachable flags for Corridor Sets
7: $c := \max(60, w_p)$
8: **if** placement for I_p not feasible w.r.t LIFO **then**
9: **if** reloading is not allowed **then return false**
10: **for placed** item $I_t \in IS$ **do**
11: **if** I_t unloaded before I_p **then continue**
12: **if** I_t is already rearranged when unloading I_p's customer **then continue**
13: **if** I_t in area $(x_p, y_p - c, 0)$ to (L, y_p, H) **then** ▷ Left Corridor
14: **if** I_t supports $I_p < 50\%$ **then**
15: $LC := LC \cup I_t$
16: **else if** $y_p - y_t \geq \lambda$ **then** r_{LC} := **false**
17: **end if**
18: **end if**
19: **if** I_t in area $(x_p, y_p, 0)$ to $(L, y_p + c, H)$ **then** ▷ Front Corridor
20: **if** I_t supports $I_p < 50\%$ **then**
21: $FC := FC \cup I_t$
22: **else if** $(x_t + l_t) - (x_p + l_p) \leq \lambda$ **then** r_{FC} := **false**
23: **end if**
24: **end if**
25: **if** I_t in area $(x_p, y_p + w_p, 0)$ to $(L, y_p + w_p + c, H)$ **then** ▷ Right Corridor
26: **if** I_t supports $I_p < 50\%$ **then**
27: $RC := RC \cup I_t$
28: **else if** $(y_t + w_t) - (y_p + w_p) < \lambda$ **then** r_{RC} := **false**
29: **end if**
30: **end if**
31: **end for**
32: **if** r_{LC} = **false and** r_{FC} = **false and** r_{RC} = **false then return false**
33: $MS := \min\{|LC|, |RC|, |FC|\}$ with corresponding reachability flag = true
34: $sum_{\text{effort}} := 0$
35: **for each** item $I_s \in MS$ **do**
36: sum_{effort} += 2· unloading effort for I_s
37: flag I_s as reloaded when unloading I_p's customer
38: **end for**
39: **elseif** Reachability Constraint Check(I_p) **then return false**
40: **end if**
41: **return true**, sum_{effort}

left and right edges must be compared for the left and right corridors, respectively. This is also visualized in Fig. V.3.

If the item I_p is not reachable, the corridor is excluded from further consideration by setting the reachability flag [16, 22, 28]. In contrast, if the item I_t can be reloaded, it is added to the corridor set [15, 21, 27]. The next step is to select the best corridor, i.e., the one that is reachable and contains the fewest elements [33]. Finally, this best set is stored in the minimal set MS. Then, the unloading effort for every item of the set MS is calculated, and the unloading flag is set to true. This flag has the effect that the unloading effort at the time of I_p's customer is only calculated once [40-43]. Finally, the unloading effort and the position's feasibility are returned [46].

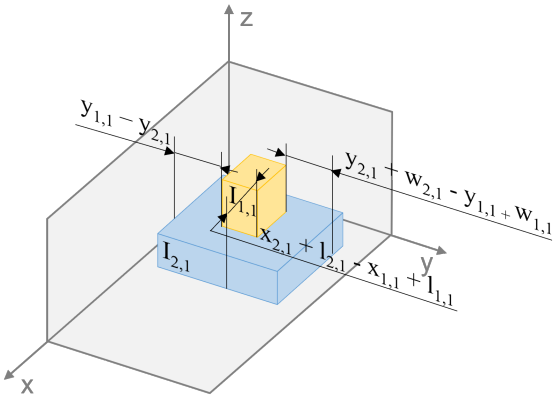


Figure V.3: Reachability of Items

6 Computational Experiments

This section investigates the solution quality, the performance of the hybrid algorithm, and the reloading effort's impact. All results and detailed packing plans are available via https://github.com/CorinnaKrebs/Results. The hybrid algorithm was implemented in C++ as a single-core x64-application and was compiled using the GCC compiler, version 4.8.3. The experiments were executed on a high-performance cluster 32-Core with 2.9 GHz.

6.1 Computational Setup

This section deals with the computational setup. It contains details concerning the objective function, computational parameters, and instance sets.

6.1.1 Unloading Effort in Objective Function

The unloading effort is given in time units. To include the unloading effort into the objective function, a more specific declaration of the speed of a vehicle is necessary to determine the total travel time. The 3L-VRPTW reflects last-mile deliveries, which is a combination of extra-urban trips (average speed is 90 km/h) and intra-urban trips (average speed is 30 km/h, see Statista (2009)). Therefore, the average speed for a vehicle v is set to 60 km/h or 1 km per minute. Consequently, the time values are given in minutes.

6.1.2 Parameters

The parameters for the loading constraints and the routing heuristic are listed in Table V.3. A preliminary study was conducted to tune the parameters for the routing heuristic. As the evaluation showed, the best results were obtained by the parameters described in Koch et al. (2018); therefore, these parameters are used for the following experiments. In addition, the parameters for the loading constraints are those established in recent literature.

6.1.3 Instances

Currently, there are three instance sets for the 3L-VRPTW: the instance sets by Moura and Oliveira (2009), Zhang et al. (2017), and Krebs et al. (2021). The characteristics of the instance sets are shown in Table V.4. However, not all instance sets are suitable for pure manual unloading as the items exceed the allowed maximum of some dimensions. For instance, the shipping conditions of DHL, one of the leading parcel delivery companies worldwide, restrict the dimensions of daily parcels to a maximal length of 120 cm and mass of 32 kg. The available instance sets all contain items exceeding these limits. Therefore, this paper presents a new instance set, accessible via https://github.com/CorinnaKrebs/Instances/Krebs_MTM.

Table V.3: 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
λ	Reachability	Minimal distance to reach an item [cm]	50

This instance set systematically varies the number of customers, items, and item types. The number of customers is between 50 and 200 in steps of 50. The customers are positioned rather close to each other to represent an urban situation. The total number of items ranges between 100 and 400 in steps of 100, thus accounting for the requirement that the number of items exceeds the number of customers. There are either 10 rather homogeneous instances or 100 item types (rather heterogeneous). For each property, there are six instances. In total, the instance set consists of 120 instances.
The vehicle parameters are based on the box truck Daily 35 by IVECO. The dimensions for the

Table V.4: Overview of Instance Sets

author	MTM applicable	#	n	m
Moura and Oliveira (2009)	x	46	25	1050, 1550
Zhang et al. (2017)	x	27	[15, 100]	[26, 199]
Krebs et al. (2021)	x	600	20, 60, 100	200, 400
This paper	✓	120	[50, 250]	[100, 400]

items are [20, 120] cm length, [20, 60] cm width, [10, 60] cm height, and they have a maximum weight of 22 kg, as MTM is not applicable for higher weights. The fragility flag is assigned randomly to the items, where approx. 10% are fragile.

6.2 Computational Results

This section evaluates the unloading effort—specifically concerning its impact on the objective value and performance. Thus, the new instance set is used. Two computational experiments are conducted: The first evaluates the algorithm variants based on predefined routes with given volume ranges. In the second, the algorithms are tested in the context of the 3L-VRPTW with the hybrid algorithm. In the following, summarized results are presented. All packing plans are available via https://github.com/CorinnaKrebs/Results. The impact of the unloading effect is determined by comparing the results with the hard LIFO constraint and when allowing reloadings (relaxed LIFO constraint) using the MTM approach.

6.2.1 Predefined Routes

This computational experiment assesses the effectiveness of the hard LIFO constraint compared to the relaxed approach, including reloadings. Therefore, predefined routes for each instance of the new instance set are created. The corresponding volume of all included items in the route is calculated and set according to the vehicle loading space. Each route is packed twice: one time with hard LIFO constraint ("only LIFO") and one time with allowing reloadings ("with reloading"). In Table V.5, the results are presented. The "Interval" column shows the used volume relative to the vehicle loading space in percent. Moreover, the total number of routes that could be packed at least once is shown, followed by a column presenting the absolute number of feasible packed routes per configuration. The column "Success Rate" gives the total number of successfully packed routes related to the total number of routes. Finally, the average runtime per volume range is presented. Routes that could not be packed at least once were excluded from this analysis.

Table V.5: Comparison with predefined routes

Volume Interval	total number of routes per interval	number of feasible packed routes		success rate [%]		avg. runtime [s]	
		only LIFO	with Reloading	only LIFO	with Reloading	only LIFO	with Reloading
40-45	50,626	47,694	50,631	94.2	99.9	1.30	1.38
45-50	60,824	50,947	60,823	83.8	99.9	1.45	1.58
50-55	59,539	39,058	58,970	65.6	99.0	1.74	1.95
55-60	51,672	24,186	48,957	46.8	94.7	1.87	2.41
60-65	33,946	11,721	30,266	34.5	89.2	1.75	2.94
65-70	17,462	3,969	15,599	22.7	89.3	1.69	3.31
70-75	7,257	707	6,747	9.7	92.9	1.69	3.76
75-80	2,735	47	2,691	1.7	98.4	1.43	4.37
80-85	838	2	835	0.2	99.6	1.22	5.13
total	284,905	178,331	281,205	39.9	95.9	1.56	2.11

The results demonstrate that allowing reloadings leads to a significantly higher success rate (1.5 times on average). This is especially true the higher the volume utilization is: While the deviation of the success rate between the hard LIFO constraint and allowing reloading is 5.7% points in the interval "40-45", it is 99.4% points in the interval "80-85". Concurrently, however, the average runtime increases due to allowing reloadings. Regarding the overall average, the runtime is 1.4 times higher when allowing reloadings than without (hard LIFO). Surprisingly, the trend of "the higher the volume utilization, the lower the success rate" only applies to the hard LIFO configuration. On the contrary, when allowing reloading, the lowest success rate is for the interval "65-70.". The reason for this effect lies in the varying number of total routes per interval and the difficulty of the predefined routes (e.g. the number of customers and the number of items).

6.2.2 Combination with Vehicle Routing

The following Table V.6 presents the average results out of 25 runs for two configurations: when the LIFO constraint is a hard constraint ("only LIFO") and when reloading is included ("with Reloading"). The deviation between both configurations is also provided. According to these results, the sum of the number of used vehicles (v_{used}), the total travel distance (ttd), the total

travel time (ttt), the unloading effort ("effort") and the average runtime in seconds ("runtime") are shown. In addition, the new instance set enables comparisons between the number of customers (n), items (m), and item types.

Table V.6: Comparison of Average Results

		n				m				Item types		Total
		50	100	150	200	100	200	300	400	10	100	
only LIFO												
	sum v_{used}	223	212	209	172	27	153	281	355	399	417	**816**
	sum ttd	3,987.66	4,172.94	4,692.24	3,694.58	713.51	3,437.59	5,793.17	6,603.13	8,136.54	8,410.87	**16,547.41**
	sum ttt	16,973.32	21,859.62	29,532.23	25,508.56	3,406.48	19,902.56	34,100.38	36,464.30	46,687.29	47,186.43	**93,873.72**
	sum effort	1,390.76	1,249.87	1,242.19	969.25	141.98	844.41	1,666.88	2,198.81	2,420.57	2,431.50	**4,852.07**
	avg. runtime [s]	3,594.50	3,600.00	3,600.00	3,600.00	3,578.00	3,600.00	3,600.00	3,600.00	3,596.33	3,600.00	**3,598.17**
with Reloading												
	sum v_{used}	196	184	173	147	21	130	240	309	342	358	**700**
	sum ttd	3,504.99	3,550.28	4,023.99	3,181.45	659.28	3,007.39	4,981.12	5,612.93	7,030.39	7,230.33	**14,260.71**
	sum ttt	15,782.55	20,606.85	28,268.68	24,590.35	3,303.34	19,089.00	32,442.79	34,413.32	44,330.12	44,918.33	**89,248.45**
	sum effort	1,302.49	1,179.08	1,164.60	918.49	133.80	798.90	1,569.22	2,062.74	2,249.76	2,314.90	**4,564.66**
	avg. runtime [s]	3,459.36	3,600.00	3,600.00	3,600.00	3,037.44	3,600.00	3,600.00	3,600.00	3,539.31	3,566.93	**3,553.12**
	diff. v_{used}	-12.11%	-13.21%	-17.22%	-14.53%	-22.22%	-15.03%	-14.59%	-12.96%	-14.29%	-14.15%	**-14.22%**
	diff. ttd	-12.10%	-14.92%	-14.24%	-13.89%	-7.60%	-12.51%	-14.02%	-15.00%	-13.59%	-14.04%	**-13.82%**
	diff. ttt	-7.02%	-5.73%	-4.28%	-3.60%	-3.03%	-4.09%	-4.86%	-5.62%	-5.05%	-4.81%	**-4.93%**
	diff. effort	-6.35%	-5.66%	-6.25%	-5.24%	-5.76%	-5.39%	-5.86%	-6.19%	-7.06%	-4.80%	**-5.92%**
	diff. runtime	-3.76%	0.00%	0.00%	0.00%	-15.11%	0.00%	0.00%	0.00%	-1.59%	-0.92%	**-1.25%**

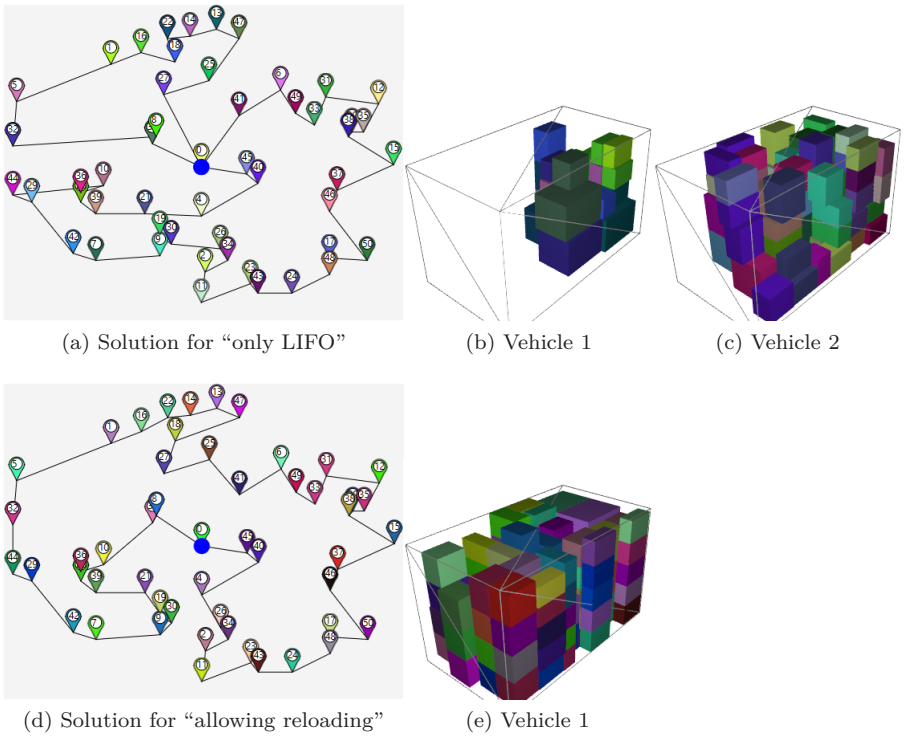
As before, the results evidence that the objective values improve significantly by allowing reloading. The additional effort caused by reloading is less than the savings from finding a shorter total travel distance. Both the total number of used vehicles and the total travel distance decrease by about 14%, on average. Concerning the total travel time, including the reloading effort, the saving is 5%. Moreover, the solutions can also be found more quickly; on average, the running time is reduced by almost 1.25%. Due to the larger solution space, more and better solutions are available so that the ALNS can terminate earlier.

Two clear trends can be identified in the difference between "only LIFO" and "allowing reloading." First, the effectiveness of reloading decreases with the number of customers: For every 50 additional customers, the difference in the total travel time decreases by about 1.5 percentage points. One reason for this tendency is that with a higher number of customers, there is an accordingly higher number of items that might be unloaded later and block other items. The actual number of items per customer is less relevant. Another correlation can be identified with the number of items. The difference in the total travel time increases along with the number of items. This is a sign of the effectiveness of reloading. The reason is that the LIFO constraint correlates negatively with the number of items; thus, the benefits of reloading increase with the number of items. For every 100 items, the difference increases by about 0.8 percentage points. The number of item types and, thus, the heterogeneity of the items have almost no influence on the total time.
One concrete example of a solution is presented in Fig. V.4, which displays a comparison of solutions for "only LIFO" and when "allowing reloading" for instance no. 5. In this case, the objective values decrease significantly: the total travel time decreases by 3%, the number of used vehicles is halved, and even the total travel distance is 10% shorter when "allowing reloading." In addition, the solution is received 40% faster.

To summarize, allowing for reloading to relax the LIFO constraint reduces the objective values significantly and lowers the runtime.

7 Conclusion

This paper presents an approach for considering the manual unloading effort in the context of the combined 3D loading and container loading problem with time windows (3L-VRPTW). For this purpose, a standard method to evaluate manual work in the industrial environment is applied: Methods-Time-Measurement (MTM). Thus, for every item, the necessary effort for its unloading is calculated and integrated into the objective function, which aims at minimizing the total travel distance. Additionally, the MTM approach is used to relax the LIFO constraint. The LIFO constraint is rather strict, as no item is allowed to be placed in front of items which are unloaded later or placed on top of them. In this situation, relaxation is achieved by allowing for items that block items. For these items, the effort of unloading and reloading is calculated and integrated into



(a) Solution for “only LIFO” (b) Vehicle 1 (c) Vehicle 2

(d) Solution for “allowing reloading” (e) Vehicle 1

Figure V.4: Comparison “only LIFO” and “allowing reloading” for instance no. 5

the objective function. Two computational experiments are conducted: One evaluates the success rate for predefined routes, and one determines the impact within 3L-VRPTW through the hybrid algorithm. For this purpose, a new instance set tailored for urban deliveries is generated, varying in the number of customers, items, and item types. The comparison is based on the results received with the hard LIFO constraint, disallowing reloadings, and the relaxed LIFO constraint, allowing reloadings. The results clearly show that the latter approach improves the results significantly, expressed by a decrease of 5% for the total travel time, on average, compared to the results with the hard LIFO constraint. Moreover, by allowing reloadings, solutions are found faster by an average of 1.25%.

To summarize, including the unloading effort into the objective function reflects the reality and allows to include reloadings, which significantly improves the solution quality and runtime.

As future work, the rearrangement of items could be improved: In this paper, blocking items are entirely unloaded and reloaded. An improved approach could be that the current packing plan inside the vehicle loading space is analyzed and the blocking item placed in a different position.

Acknowledgements. The author declares that she has no conflict of interest.

References

Bischoff, E. E. and Ratcliff, M. S. W. (1995). “Issues in the development of approaches to container loading”. In: *Omega* vol. 23, no. 4, pp. 377–390. ISSN: 03050483. DOI: 10.1016/0305-0483(95)00015-G.

Ceschia, S., Schaerf, A., and Stützle, T. (2013). “Local search techniques for a routing-packing problem”. In: *Computers and Industrial Engineering* vol. 66, no. 4, pp. 1138–1149. ISSN: 0360-8352. DOI: https://doi.org/10.1016/j.cie.2013.07.025. URL: http://www.sciencedirect.com/science/article/pii/S0360835213002404.

Clarke, G. and Wright, J. W. (1964). “Scheduling of Vehicles from a Central Depot to a Number of Delivery Points”. In: *Operations Research* vol. 12, no. 4, pp. 568–581. ISSN: 0030364X, 15265463. DOI: 10.1287/opre.12.4.568. URL: http://www.jstor.org/stable/167703.

Gajda, M., Trivella, A., Mansini, R., and Pisinger, D. (2022). "An optimization approach for a complex real-life container loading problem". In: *Omega* vol. 107, p. 102559. ISSN: 0305-0483. DOI: https://doi.org/10.1016/j.omega.2021.102559. URL: https://www.sciencedirect.com/science/article/pii/S0305048321001687.

Gendreau, M., Iori, M., Laporte, G., and Martello, S. (2006). "A Tabu Search Algorithm for a Routing and Container Loading Problem". In: *Transportation Science* vol. 40, no. 3, pp. 342–350. ISSN: 0041-1655. DOI: 10.1287/trsc.1050.0145. URL: http://pubsmisc.informs.org/doi/abs/10.1287/trsc.1050.0145.

Junqueira, L., Oliveira, J. F., Carravilla, M. A., and Morabito, R. (2013). "An optimization model for the vehicle routing problem with practical three-dimensional loading constraints". In: *International Transactions in Operational Research* vol. 20, no. 5, pp. 645–666. DOI: https://doi.org/10.1111/j.1475-3995.2012.00872.x. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1475-3995.2012.00872.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1475-3995.2012.00872.x.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). "Optimization by simulated annealing." In: *Science* vol. 220 4598, pp. 671–80. DOI: 10.1126/science.220.4598.671.

Koch, H., Bortfeldt, A., and Wäscher, G. (Feb. 2018). "A hybrid algorithm for the vehicle routing problem with backhauls, time windows and three-dimensional loading constraints". In: *OR Spectrum* vol. 40. DOI: 10.1007/s00291-018-0506-6.

Krebs, C., Ehmke, J. F., and Koch, H. (2021). "Advanced loading constraints for 3D vehicle routing problems". In: *OR Spectrum*. ISSN: 1436-6304. DOI: 10.1007/s00291-021-00645-w. URL: https://doi.org/10.1007/s00291-021-00645-w.

— (2023). "Effective loading in combined vehicle routing and container loading problems". In: *Computers & Operations Research* vol. 149, p. 105988. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2022.105988. URL: https://www.sciencedirect.com/science/article/pii/S0305054822002258.

Moura, A. (2008). "A Multi-Objective Genetic Algorithm for the Vehicle Routing with Time Windows and Loading Problem". In: *Intelligent Decision Support: Current Challenges and Approaches*. Ed. by Bortfeldt, A., Homberger, J., Kopfer, H., Pankratz, G., and Strangmeier, R. Wiesbaden: Gabler, pp. 187–201. ISBN: 978-3-8349-9777-7. DOI: 10.1007/978-3-8349-9777-7. URL: https://doi.org/10.1007/978-3-8349-9777-7.

Moura, A. and Oliveira, J. F. (Oct. 2009). "An integrated approach to the vehicle routing and container loading problems". In: *OR Spectrum* vol. 31, no. 4, pp. 775–800. ISSN: 1436-6304. DOI: 10.1007/s00291-008-0129-4. URL: https://doi.org/10.1007/s00291-008-0129-4.

MTM Association e.V. (2020). URL: https://mtm.org/en/brands/brand-history.

Reil, S., Bortfeldt, A., and Mönch, L. (2018). "Heuristics for vehicle routing problems with backhauls, time windows, and 3D loading constraints". In: *European Journal of Operational Research* vol. 266, no. 3, pp. 877–894. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2017.10.029. URL: http://www.sciencedirect.com/science/article/pii/S0377221717309426.

Rückert, P., Papenberg, B., and Tracht, K. (2021). "Classification of assembly operations using machine learning algorithms based on visual sensor data". In: *Procedia CIRP* vol. 97. 8th CIRP Conference of Assembly Technology and Systems, pp. 110–116. ISSN: 2212-8271. DOI: https://doi.org/10.1016/j.procir.2020.05.211. URL: https://www.sciencedirect.com/science/article/pii/S2212827120314311.

Statista (2009). URL: https://de.statista.com/statistik/daten/studie/37200/umfrage/durchschnittsgeschwindigkeit-in-den-15-groessten-staedten-der-welt-2009/.

Tarantilis, C. D., Zachariadis, E. E., and Kiranoudis, C. T. (June 2009). "A Hybrid Metaheuristic Algorithm for the Integrated Vehicle Routing and Three-Dimensional Container-Loading Problem". In: *IEEE Transactions on Intelligent Transportation Systems* vol. 10, no. 2, pp. 255–271. ISSN: 1524-9050. DOI: 10.1109/TITS.2009.2020187.

Zhang, D., Cai, S., Ye, F., Si, Y.-W., and Nguyen, T. T. (2017). "A hybrid algorithm for a vehicle routing problem with realistic constraints". In: *Information Sciences* vol. 394-395, pp. 167–182. ISSN: 0020-0255. DOI: 10.1016/j.ins.2017.02.028.

Appendix

A Removal and Insertion Operators

Table V.7 shows nine removal operators, and Table V.8 summarises the three insertion approaches used in this paper. This paper uses the removal and insertion operators described and evaluated in Koch et al. (2018).

Table V.7: Overview Removal Operators

Neighborhood 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 them
Neighbour graph	Removes customers, increasing the average distance of a tour
Overlap	Removes customers, leading to the intersection of two tours
Inner Route	Removes a tour surrounded by another, splits the surrounding tour into two
Intersection	Removes customers, leading to intersections within a tour
Tour Pair	Removes two intersecting tours

Table V.8: Overview Insertion Operators

Neighborhood Operators	**Description**
Greedy	Inserts customers iteratively so that an increase in routing costs is minimal
Regret-2	Inserts customers iteratively so that the maximal difference of routing costs for the best and second-best insertion in different tours is achieved
Regret-3	Inserts customers iteratively so that the sum of two differences in 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

Paper VI

Solution Validator and Visualizer for (Combined) Vehicle Routing and Container Loading Problems

Corinna Krebs, Jan Fabian Ehmke

Published in *Annals of Operations Research*, May 2023, volume 326, pp. 561–579. DOI: 10.1007/s10479-023-05238-0.

Abstract

The optimization of cargo loading and transportation are two highly considered optimization problems (namely “3L-CVRP"). The combination of both has attracted increasing interest in the past decades. Hereby, 2D or 3D items have to be transported from one depot to a given set of customers using a homogeneous fleet of vehicles. Each route must be provided with a feasible packing plan taking various constraints into account. Combining the two optimization problems increases the complexity of the solution approaches, leading to a higher difficulty to check the results for correctness.

To support the research progress and to enable transparency of solution structures, this paper provides an overview of recent literature, problem formulations, and best-known solutions. Furthermore, we introduce two open-source tools: The “Solution Validator" checks the feasibility of solutions in terms of considered constraints. The “Visualizer" provides two views and visualizes solutions. In the *vehicle routing view,* the tour plan and the corresponding schedule are displayed. In the *loading view,* the position of each item in the cargo space is demonstrated. In both views, it is possible to check the feasibility of the solution and highlight violated items. Besides the combined problem, the tool can be used also for one optimization problem (e.g. vehicle routing problem or container loading). The source codes for both tools are available at GitHub in C++ and Java and can be easily integrated into other researchers’ code.

Contents

1 Introduction

Since its introduction by Gendreau et al. (2006), the combined Vehicle Routing (VRP) and Container Loading Problems (CLP) have consistently challenged researchers worldwide. The 3L-CVRP assumes the delivery of 3D 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 a specific set of loading constraints. An exemplary solution for a 3L-CVRP instance is provided in Fig. VI.1.

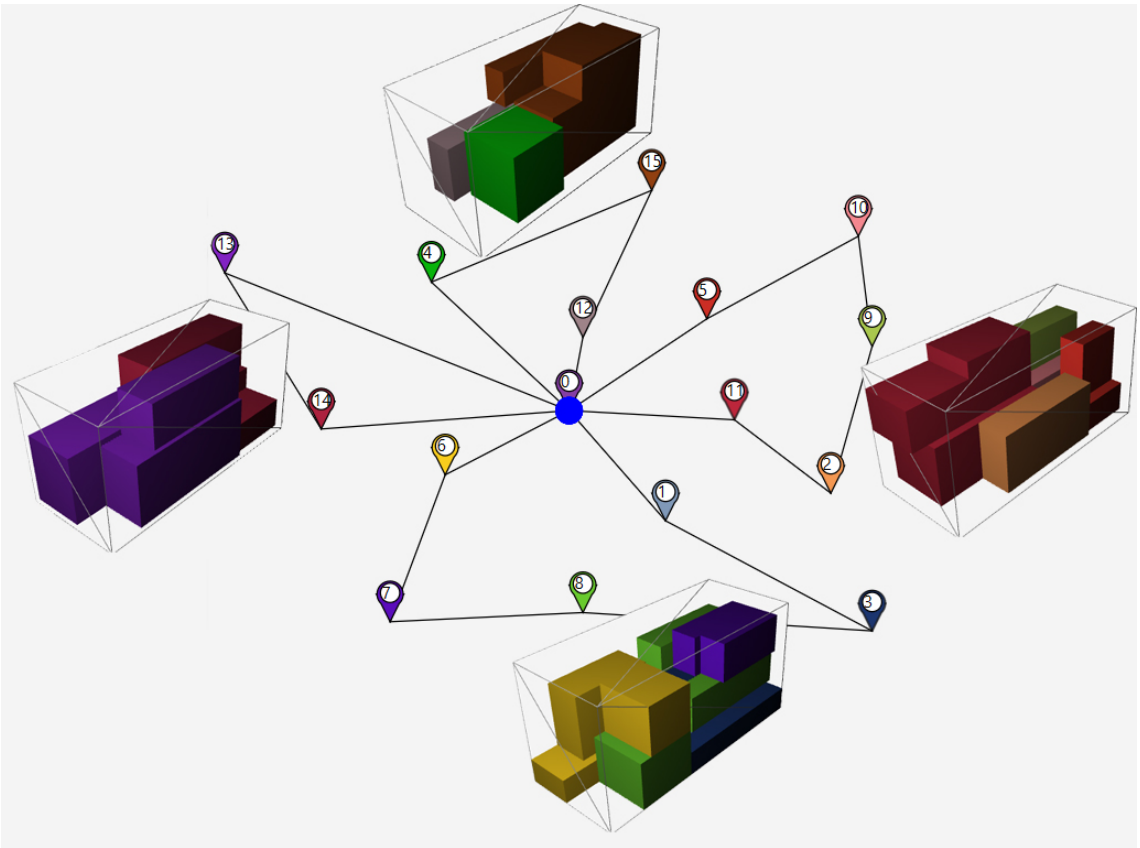


Figure VI.1: Exemplary solution for instance “3l-cvrp01"

An extension of the 3L-CVRP is the 3L-VRPTW, in which time windows at the depot and at the delivery are considered. Through the combination of the VRP with 3D CLP, the solution approaches are more complex but also more practicable. However, at the same time, the difficulty of ensuring the solution feasibility increases. This is also due to new formulations of advanced loading constraints in the latest research. These include, but are not limited to, the consideration of axle weights of vehicles (see Krebs and Ehmke 2021a), the unloading sequence, the reachability (see Ceschia et al. 2013), and load-bearing strength of items (see Krebs et al. (2021)).
This paper introduces two open-source tools to support the research for VRP, CLP, and its combined

problems[1]. The source codes of both tools are published online at GitHub and have been coded in Java and C++. The first tool, called “Solution Validator", reports the feasibility of solutions. Hereby, several constraints, especially loading constraints, can be checked concerning feasibility. In the case of infeasible solutions, every violated constraint is shown. The second tool “Visualizer" visualizes, as the name implies, the solutions. It provides two views, showing the execution of the tours and giving a detailed schedule indicating travel, waiting, and handling times. In the second view, the position of each item inside the vehicle loading space is presented. Besides that, the “Visualizer" contains an interface to the “Solution Validator" and can check the feasibility of solutions. In the case of infeasibility, the tool highlights the violated elements.

These tools support the transparency of solution structures on the one hand and further research on the other. Regarding the transparency of solution structures, during our research concerning best-known solutions, we faced several challenges: Firstly, in most papers, published results often report only the objective function value but not the vehicle tours or the coordinates of the items inside the trucks so that the correctness of the solutions cannot be guaranteed. Secondly, in rare cases, we found detailed solutions. Then, some of the solutions were either infeasible or the objective value of the solution differed from the objective value published in the research paper. Therefore, the publication of full solutions should be standard. This guarantees the correct comparison of solutions and decreases frustration caused by incorrect benchmarks.

Regarding supporting further research, first, the tool can be integrated directly into the programming code as the source code is fully published in C++ and Java. Consequently, it is not necessary to program the constraint checks. Therefore, the effort of programming new algorithms is decreased. Secondly, it is possible to visualize each processing step of a new algorithm by integrating the tools. This enables a better understanding of the algorithm so that improvements can be found and implemented. Thirdly, the “Solution Validator" checks the correctness of the solutions, which can help to detect errors in algorithms. Lastly, through visualizing the final solutions, further improvement potentials can be identified. This helps improve the solution quality (the total travel distance and/or the total time).

The paper is structured as follows: Relevant literature is reviewed in Sec. 2. To introduce the 3L-CVRP and show the covered features, the problems are formulated in Sec. 3. In Sec. 4 both tools are introduced and their application is explained. In Sec. 5, we present an overview of available instance sets, their properties, and the best-known solutions in the literature. Finally, Sec. 6 provides a summary and avenues for future work.

2 Literature Review

In the following, relevant literature with a focus on constraints for the combined vehicle routing and container loading problem is presented. In Iori et al. (2007), the Vehicle Routing Problem with Two-Dimensional Loading Constraints (2L-CVRP), a combination of the Capacitated Vehicle Routing Problem and the 2D Container Loading Problem, is introduced. To solve the optimization problem, an exact approach, based on a branch-and-cut algorithm, is provided. The three-dimensional variant, namely 3L-CVRP, is introduced by Gendreau et al. (2006). The presented solution algorithm consists of several parts: The customer sequence is determined by an “outer" Tabu Search. Then, an “inner" Tabu Search deals with the item sequence. 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). As loading constraints, the following are considered: items are packed orthogonally into the vehicle loading space (Orthogonality constraint) without overlapping through respecting their dimensions (Geometry); rotation of items only along the width-length plane (Rotation constraint); respecting the maximum vehicle’s capacity (Load Capacity); considering the fragility of items; stacking stably through requiring the support of a certain percentage of the base area (Minimal Supporting Area constraint) and unloading done by direct movements parallel to the length of the vehicle (LIFO constraint). This constraint set is here defined as *a basic constraint set* as it is commonly considered in related research. For testing, Gendreau et al. (2006) developed 27 instances.

[1]The tools are suitable for e.g. CVRP, VRPTW, SDVRP, 2L-CVRP, 3L-CVRP, 2L-VRPTW, 3L-VRPTW, 2D-CLP, 3D-CLP, SDVRP, 2D-SDVRP and 3D-SDVRP.

The 3L-CVRP has been studied intensively in recent years so that the solutions for this instance set have been improved repeatedly (e.g. Tarantilis et al. (2009), Fuellerer et al. (2010), Bortfeldt (2012), Escobar-Falcon et al. (2016) and Z. Zhang et al. (2015)). In Fuellerer et al. (2010), an extended instance set for the 3L-CVRP is generated. Tarantilis et al. (2009) present a new variant – the Capacitated Vehicle Routing Problem with Manual 3D Loading Constraints (M3L-CVRP). In contrast to the LIFO policy, it is here allowed that items hang over others (MLIFO). This adaption is also examined in a paper by Ceschia et al. (2013). Furthermore, they consider the reachability of an item. The reachability constraint was initially developed by Junqueira et al. (2013) in the context of the Three-Dimensional Bin Packing Problem, to avoid the driver standing on items to reach other items for unloading operations. Moreover, Ceschia et al. (2013) consider a robust stacking of items and the Load Bearing Strength constraint, which was first mentioned by Bischoff and Ratcliff 1995 and examined in Bischoff (2003) for the Three-Dimensional Bin Packing Problem. The Split Delivery Vehicle Routing Problem with three-dimensional loading constraints (3L-SDVRP) is included in Ceschia et al. (2013). It enables the possibility to split the demand of customers over two or more tours is, i.e. a customer can be visited several times. This problem variant is further investigated in Bortfeldt and Yi (2020), where among others the instance set by Ceschia et al. (2013) is used. D. Zhang et al. (2017) introduce 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 Moura (2008) and Moura and Oliveira (2009) the VRTWLP is introduced. This problem variant is the 3L-VRPTW without the consideration of masses (Load Capacity constraint), without the Fragility constraint, with higher Stability requirements (full support) and with more rotation possibilities. Pace et al. (2015) examine the distribution of fibre boards. For this purpose, a 70%-30% left/right balance should be obeyed. This constraint is further examined in Krebs et al. (2021). Other approaches for ensuring balanced loading within the loading space are integrated in the algorithm itself. In Ramos et al. (2017), the algorithm includes vehicle specific Load Distribution Diagrams (LDDs) that define the feasibility domain for the location of the center of gravity of the cargo. It is based on multi-population biased random-key genetic algorithm with a specialized fitness function.
Formulas for the consideration of axle weights are introduced in Krebs and Ehmke (2021a) and examined for the 2L- and 3L-CVRP. In Krebs et al. (2021), various loading constraints are analyzed and combined for the 3L-VRPTW. Moreover, new formulations are introduced: a new variant for robust stacking of items, the consideration of load-bearing strength based on the science of statics, and balanced loading inside the loading space. Further formulations for stable stacking based on the science of statics are evaluated in Krebs and Ehmke (2021b). All mentioned loading constraints can be tested w.r.t. feasibility in the “Solution Validator" tool.

3 Problem Formulation

To provide an introduction to the problem and to demonstrate the scope of the tools, the 3L-CVRP and its extension, the 3L-VRPTW, are formulated in the following by adapting the convention as presented by Koch et al. (2018). All constraints covered by the tools are briefly presented. The loading constraints are described in detail in Krebs et al. (2021) and Krebs and Ehmke (2021b).

3.1 3L-CVRP

Let $G = (N, E)$ be a complete, directed graph, where N is the set of $n+1$ nodes including one 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.
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 $h_{i,k}$. Depending on the constraints (see below), additional parameters are necessary. The items are delivered by v_{max} available, homogeneous vehicles. Each vehicle has a maximum load capacity D and a cuboid loading space defined by length L, width W and height H.

The number of used vehicles in a solution is described by v_{used} ($v_{used} \leq v_{max}$). A solution is a set of v_{used} pairs of routes R_v and packing plans PP_v ($v = 1, ..., v_{used}$). Hereby, the route R_v 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. The 3L-CVRP aims at determining a feasible solution minimizing the total travel distance ttd, and meeting all included constraints.

3.2 3L-VRPTW

In the extension with time windows ("3L-VRPTW"), three times are assigned 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. 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. The objective function is either the minimization of the total travel distance or a combination of minimizing the number of used vehicles (v_{used}) first and total travel distance second (see e.g. Moura (2008)).

3.3 Constraints

The following constraints are categorized in solution constraints (S), in routing constraints (R), and loading constraints (C). In terms of the loading constraints, there are several alternative constraint formulations for the Unloading Sequence, Vertical Stability, and Stacking. This means, only one of these alternative constraint formulations can be included into the model. All described constraints are covered in the "Solution Validator" tool.
A solution is feasible if

(S1) All routes R_v and packing plans PP_v are feasible (see below);

(S2) The number of used vehicles v_{used} does not exceed the number of available vehicles v_{max};

(S3) Each solution contains all demanded items once and all customers.

A route R_v must meet at least the following routing constraint:

(R1) Each route starts and terminates at the depot and visits at least one customer.

In case of split deliveries are not allowed, the following routing constraints must be obeyed:

(R2) Each customer is visited exactly once;

(R3) Each packing plan PP_v contains all c_i items of all customers i included in the corresponding route ($i \in R_v$).

In terms of the 3L-VRPTW, an additional routing constraint must apply:

(R4) 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. As described before, if several formulations for a loading constraint are available (e.g. Unloading Sequence), then only one formulation can be included into the optimization problem. For constraints differing from the basic constraint set by Gendreau et al. (2006), we provide the corresponding reference.

(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 item can be rotated.

(a) *Length-Width*: The items can be rotated 90° only on the width-length plane;

(b) *All Rotations*: The items can be rotated along all planes.

(C4) *Load Capacity*: The sum of masses of all included items of a vehicle does not exceed the maximum load capacity D.

(C5) *Unloading Sequence*: The items can be unloaded by movements parallel to the x-axis.

(a) *LIFO*: No item is placed above or in front of item $I_{i,k}$, which belongs to a customer served after customer i;

(b) *MLIFO by Tarantilis et al. (2009)*: No item is placed on or in front of item $I_{i,k}$, which belongs to a customer served after customer i;

(C6) *Vertical Stability*: The items are stable packed and cannot topple.

(a) *Minimal Supporting Area*: Each item has a supporting area of at least a percentage α of its base area;

(b) *Top Overhanging by Krebs et al. (2021)*: Only the topmost item of a stack is allowed to overhang obeying the Minimal Supporting Area;

(c) *Multiple Overhanging by Ceschia et al. (2013)*: The Minimal Supporting Area must be obeyed at each level of a stack;

(d) *New Static Stability by Krebs and Ehmke (2021b)*: The center of gravity of each item must be supported at each level of the stack and the Minimal Supporting Area must be obeyed;

(e) *Static Stability by Mack et al. (2004)*: The center of gravity of each item must be supported by the directly underlying items and the Minimal Supporting Area must be obeyed between each item and the indirectly supporting items laying on the ground;

(C7) *Stacking*: The items are stacked respecting their bearing capacities.

(a) *Fragility*: A fragility flag $f_{i,k}$ is assigned to each item $I_{i,k}$ to divide them into fragile items ($f_{i,k} = 1$) and non-fragile ones ($f_{i,k} = 0$). No non-fragile items are placed on top of fragile items;

(b) *Load Bearing Strength – Simplified Selection by Bischoff and Ratcliff (1995)*: Each item $I_{i,k}$ can support a maximum load per area described by parameter $lbs_{i,k}$. It must not be exceeded anywhere on the top face of an item. For the load distribution all items underneath the bottom surface are used;

(c) *Load Bearing Strength – Complete Selection by Krebs et al. (2021)*: As before, each item is assigned the $lbs_{i,k}$ parameter. The load of an item $I_{i,k}$ is distributed recursively from its bottom surface to its directly underlying items;

(C8) *Reachability by Junqueira et al. (2013)*: The distance between the front side of an item and the nearest possible position of the operator must be less or equal than a certain length λ;

(C9) *Axle Weights by Krebs and Ehmke (2021a)*: Each vehicle is assigned the wheelbase WB (distance between two axles), the length between the front axle and the loading space (L_f), and permissible axle weights FA_{perm} and RA_{perm}, which are not allowed to be exceeded;

(C10) *Balanced Loading by Pace et al. (2015)*: The mass of an item is proportional distributed to the horizontal vehicle halves according to its position. The load of one vehicle half must not exceed a certain percentage p of D.

4 Open Source Tools

Both “Solution Validator" and “Visualizer" are open source and published online via GitHub repositories. Moreover, the tools are written in Java, requiring at least version 10. For the “Solution Validator", the code is additionally available in C++ (min. version C++11). For the C++ code, no additional libraries are required. In terms of Java code, several dependencies must be provided. Therefore, Apache Maven is used to simplify the building process. The necessary *pom* file is available online along with the source code. The “Visualizer" requires the JavaFX Framework, which must be downloaded and linked separately. In the following, the necessary data format and the application are shown.

4.1 Data Format

To provide the necessary data for the tools, three files are required: An *Instance*, a *Solution*, and a *Constraint File*. All three files are explained and shown in the following. Example files are provided in the appendix.

4.1.1 Instance File

The *Instance File* contains all relevant data as described in Sec. 3. Consequently, it has information about the problem (3L-CVRP or 3L-VRPTW), nodes (coordinates, demanded items), vehicle (number of available vehicles, dimensions, parameters), and items (dimensions, parameters). An exemplary, shortened file is shown in Fig. VI.6 in the appendix. For common benchmarks, *Instance Files* are available via https://github.com/CorinnaKrebs/Instances. Moreover, the Solution Validator contains a "Write" class to create your own *Instance Files*. Another option is to use the class constructors to provide the necessary data within the program.

4.1.2 Constraint File

The *Constraint File* defines all included loading constraints and their necessary parameters (see Sec. 3.3). A complete *Constraint File* defining the *basic constraint set* by Gendreau et al. (2006) is presented in Fig. VI.7 in the appendix. Further constraint sets are published along with the solutions via https://github.com/CorinnaKrebs/Results. For own files, the “Solution Validator" provides a “Write" class, or alternatively, class constructors can be used to define the necessary data for the program.

4.1.3 Solution File

The *Solution File* contains the Routing R_v and the Packing Plans PP_v for each vehicle v. It shows the sequence of each visited customer and the exact position of each item inside the loading space. An exemplary, shortened file is shown in the appendix in Fig. VI.8. The results of our previous work are available via https://github.com/CorinnaKrebs/Results. As before, the Solution Validator includes a “Write" class that may be used to build custom *Solution Files*. Another way is to input the necessary data within the program using the class constructors.

4.2 Solution Validator

As the name implies, the Solution Validator validates the feasibility of solutions w.r.t. the observance of constraints. It is available via https://github.com/CorinnaKrebs/SolutionValidator. In the following, the scope and application are described.

4.2.1 Scope

The Solution Validator is created to check the feasibility of solutions for the combined VRP and CLP (“3L-CVRP" and “3L-VRPTW"). It checks the observance of routing and loading constraints in each step of each tour. Hereby, a subset of loading constraints (as described in the Problem Formulation in Sec. 3) can be used for the feasibility check.
The Solution Validator can also be adapted to check the feasibility for solely the VRP or CLP

as shown below. The Problem Formulation in Sec. 3 defines the features of this tool: In terms of the VRP, this tool can deal with one depot where the demand is delivered by a homogeneous fleet without a split of the delivery. Time Windows at the depot and at customer locations can be obeyed optionally. Regarding the CLP, it is also possible to adapt it for the 2D case by setting the height of each item to the cargo loading space height H.

4.2.2 Application

In the following Code 1, the Java Program is presented. The code is structured in three parts: reading data from files, a feasibility check, and notification about the feasibility check. The C++ Code is structured in a similar way. In lines 2 to 4, the *Instance*, *Constraint*, and the *Solution File* are read. Hereby, the necessary paths (*pathToInstanceFile*, *pathToConstraintFile*, *pathToSolutionFile*) to the files must be provided as strings. Instead of fixed strings for the definition of the file paths, one can adapt lines 2 to 4 so that the program arguments (args) can be used to inject the file paths which is shown in Code 2. In the next part (feasibility check), the solution is checked w.r.t. routing constraints (line 5) and loading constraints (line 6). Internally, only the activated constraints as specified in the *Constraint File*, are checked. In the last part, a message is printed depending on the feasibility and the program exits returning an exit code indicating the feasibility status.

Code 1 Original Java Solution Validator Program

```
1    public static void main(String[] args) {
2        Instance instance = Read.readInstanceFile(pathToInstanceFile);
3        ConstraintSet constraintSet =
         ↪  Read.readConstraintFile(pathToConstraintFile);
4        Solution solution = Read.readSolutionFile(pathToSolutionFile,
         ↪  instance);
5        if (checkRoutingConstraints(solution, constraintSet, instance, true)
6            && checkLoadingConstraints(solution, constraintSet, instance,
             ↪  true)) {
7            System.out.println("All Constraints checked. Solution is
             ↪  feasible.");
8            System.exit(1);
9        }
10       System.err.println("Solution is not feasible. Please check error hints
         ↪  above.");
11       System.exit(-1);
12   }
```

Code 2 Java Solution Validator Program using Program Arguments

```
1    public static void main(String[] args) {
2        Instance instance = Read.readInstanceFile(args[0]);
3        ConstraintSet constraintSet = Read.readConstraintFile(args[1]);
4        Solution solution = Read.readSolutionFile(args[2], instance);
5        ...
```

The Solution Validator can be adapted to checking only the VRP by removing the Loading Constraints Check in line 6. For the check of the CLP, the Routing Constraints Check in line 5 must be removed respectively. Further adaption can be implemented easily as the entire source code is structured and well documented.

4.3 Visualizer

The “Visualizer" creates interactive views of the 3L-CVRP and 3L-VRPTW solutions. The tool is published online via https://github.com/CorinnaKrebs/Visualizer. In the following, the tool is presented in more detail.

4.3.1 Scope

The Visualizer displays the solution of Vehicle Routing and Container Loading Problems in separated views. It has the same limitations as the Solution Validator (see Problem Formulation, Sec. 3). The tool enables further analysis of solutions: In terms of the VRP, the tool shows the resulting tours including their distances and the total time per tour. Thus, improvement potentials can be identified. Regarding the CLP, the loading space and the loading sequence are visualized. This assists in understanding the loading process and determining weaknesses of placements (e.g. unbalanced or unstable). Moreover, the Visualizer has an interface to the Solution Validator so that the feasibility of each solution can be checked. Infeasible elements (tours or items) are directly highlighted. This is beneficial e.g. to trace errors in the solution approach.

4.3.2 Application

The Visualizer has three main views: one for the data input, one for the VRP, and one for the CLP. The application of each view is described in the following.

After executing the Visualizer, a welcome view appears. By clicking on the *Start* menu on *File Open*, one reaches the data input view (see Fig. VI.2).

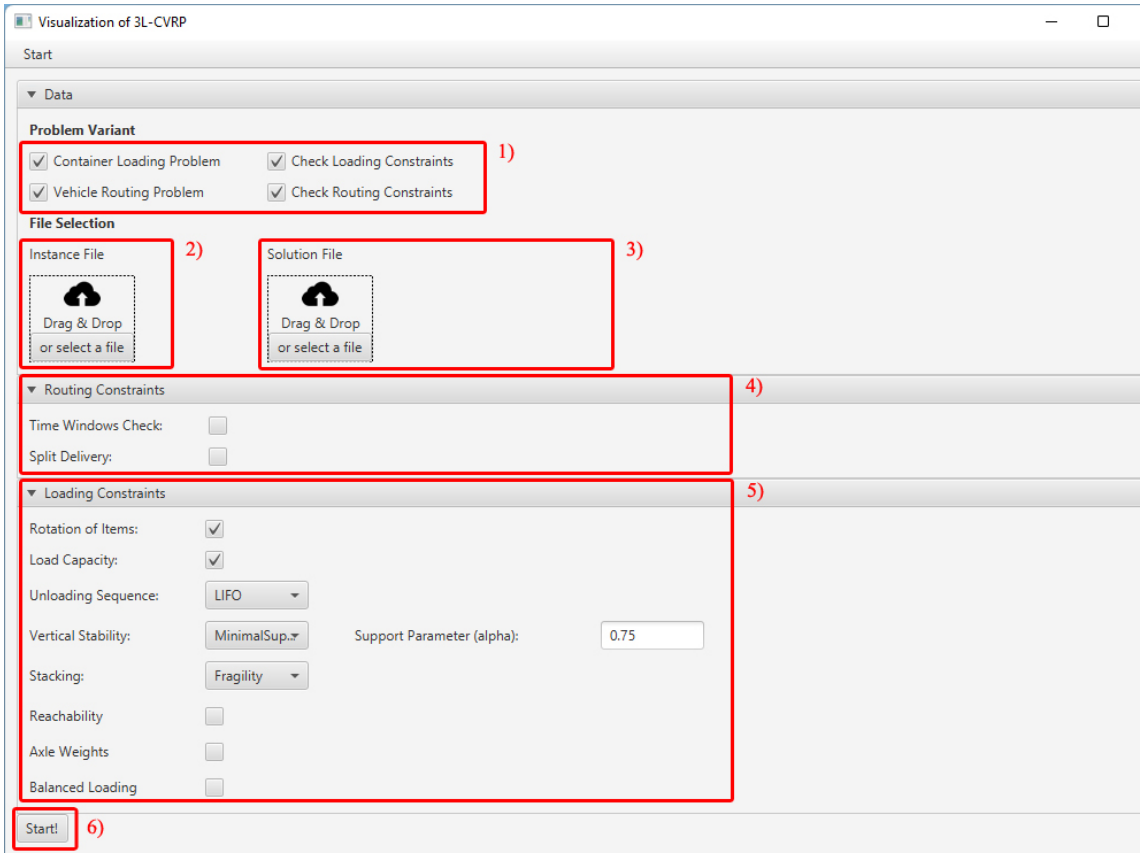


Figure VI.2: View of the Data Input Mask

In the following, the areas shown in Fig. VI.2 are described in more detail:

VI.2.1 In the left column, the problem is defined and the views are activated or deactivated accordingly. In the right column, one can (de-)activate the constraints check and therefore, the interface to the Solution Validator.

VI.2.2 The next field is to provide the *Instance File*. It is possible to Drag & Drop the file directly over the dotted field. Alternatively, one can select the file via a File Browser. After providing the *Instance File*, the field changes as in the next item. Then, the path to the file is shown.

VI.2.3 In this area, the *Solution File* can be provided. As the *Solution File* is already selected, the path is shown. The field can be reset by clicking on the cross button.

VI.2.4 In the center area of this view, additional routing constraints for the feasibility check can be included.

VI.2.5 In the bottom area of this view, it is possible to define the subset of the loading constraints for the feasibility check. If one loading constraint has several formulations as described in Sec. 3, a drop-down list for the selection appears.

VI.2.6 If the necessary data is provided, the start button can be clicked. The current view is then closed and the Solution views (CLP and/or VRP) are opened. This might take some time due to the feasibility check.

The VRP view is exemplarily shown in Fig. VI.3. Directly after opening, an animation starts showing the complete routing process.

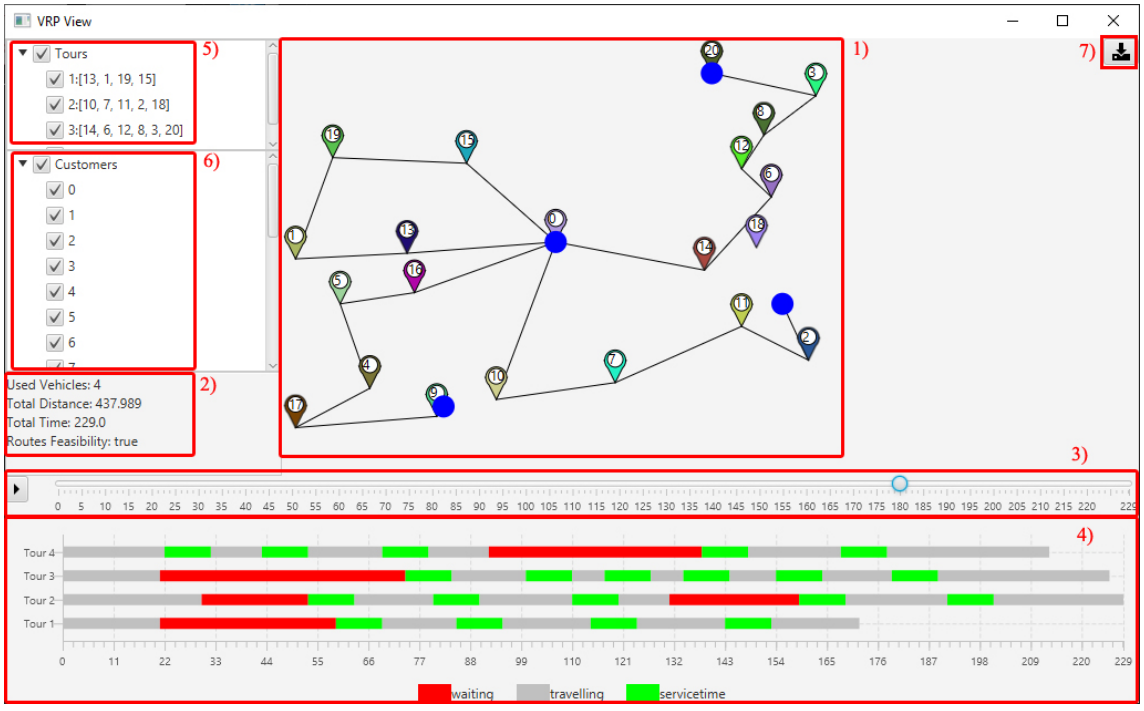


Figure VI.3: View of the Vehicle Routing Problem

Regarding Fig. VI.3, the areas describe the following:

VI.3.1 In the center of the view, the depot and the customer locations are displayed. Starting from the depot, the tours are shown. The vehicle is indicated as a blue circle. Each customer has its unique color. Its corresponding items use the same color in the CLP view. Within this area, it is possible to zoom in or out via the mouse wheel in order to see further details.

VI.3.2 In this field, general information about the solution is provided, such as the total number of used vehicles, the total travel distance, and the total time. Moreover, if activated, the feasibility of the routes is shown.

VI.3.3 Through the slider, it is possible to jump to each step of the routing process (0 to total time). Consequently, it enables tracking the position and status of each vehicle within the tour at each timestamp. The play button starts the animation which automatically goes through each step of the routing process.

VI.3.4 The underlying Gantt chart displays the current vehicle status per tour and per timestamp. Hereby, three vehicle statuses exist: The traveling time, the service time (unloading time), and the waiting time that occurs when the vehicle has to wait until the start time. As the waiting time is not value-adding in the process, the tool helps to identify and then minimize waiting times.

VI.3.5 This tree enables changing the visibility of tours. This might be helpful in case of hidden details.

VI.3.6 As in the previous item, this tree changes the visibility of the customer pins.

VI.3.7 Through the download button, it is possible to download the currently displayed area for further research purposes.

In Fig. VI.4, the CLP view is presented. Similar to the VRP view, an animation showing the loading process starts directly after opening the view.

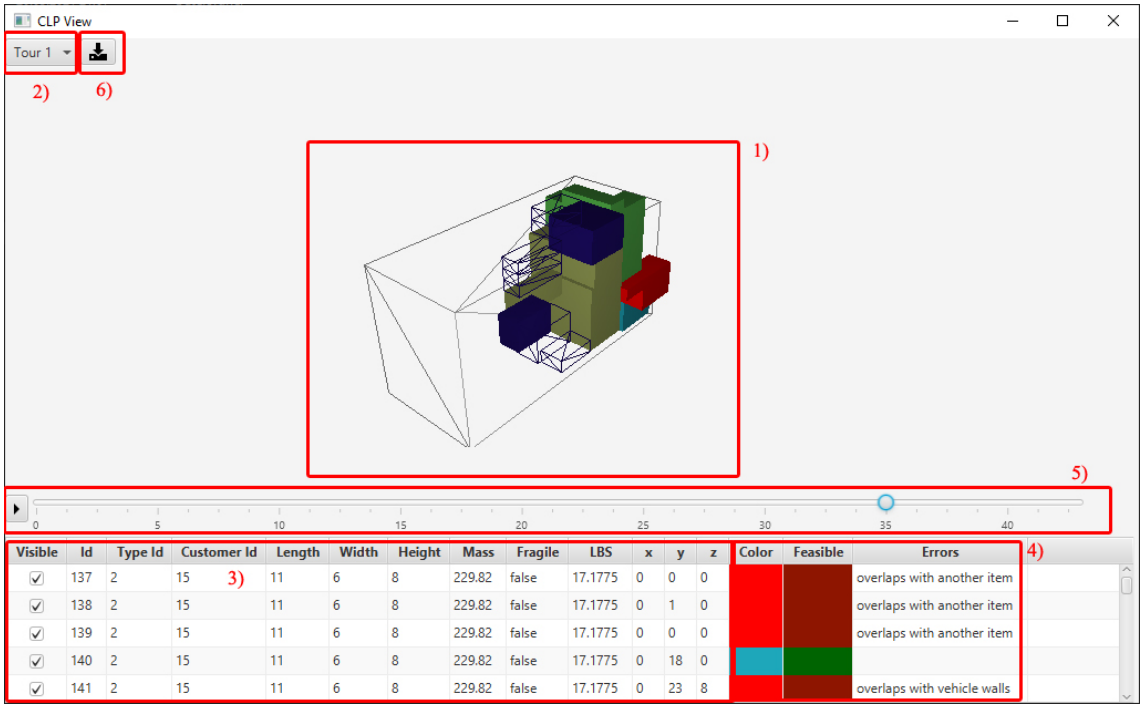


Figure VI.4: View of the Container Loading Problem

As shown in Fig. VI.4, there are six areas in the CLP view:

VI.4.1 In the center, the loading space of the first tour is displayed. It shows the position of each item inside the loading space. The loading space can be completely rotated and scaled to analyze the positions effectively. When double-clicking on an item, the corresponding row in the table gets highlighted.

VI.4.2 Through the drop-down menu, it is possible to switch the tours and consequently, to change the loading space in the center.

VI.4.3 In the underlying table, the information about the items is presented. The visibility of each item can be changed in the first column. An invisible item is shown via its borders in the loading space. By clicking on the header, the table can be sorted according to the clicked column.

VI.4.4 The last three columns indicate the feasibility of the item position. If the position of an item is not feasible, its color changes from the customer's color to red. The feasibility status is indicated in the column "Feasible" through green for feasible and red for infeasible. In the last column, an error text describes the violated constraint.

VI.4.5 As in the VRP view, there is also a slider to visualize the loading process per each loaded item. This enables the possibility to analyze the sequence of the loading processes and identify inefficient item positions. Through the play button, the animation can be started showing the loading of items step by step.

VI.4.6 The download button enables downloading a picture of the displayed loading space with its current loading step.

4.3.3 Code Structure

If it becomes necessary to modify the source code, it is beneficial to understand the structure of the code. Hereby, the Model-View-Presenter (MVP) design as proposed by Potel (1996) is used (see Fig. VI.5).

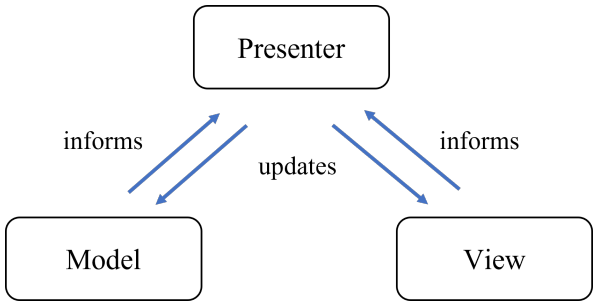


Figure VI.5: MVP Design

The code is distributed in three parts: The model, the view, and the presenter. The model contains all the necessary data. The view is responsible to display the data via components. It has no direct link to the model. The presenter connects the model and the view. It updates the components of the view in case of data changes in the model, and it updates also the model according to user inputs in the view. The presenter is informed through internal events to trigger updates if needed. For each view (Main view, CLP view, VRP view) the MVP design is implemented. Moreover, it is also applied to layout components within the views (e.g. text fields).

5 Instances and Best Known Results

In this section, we first present common instance sets for the 3L-CVRP and the 3L-VRPTW with their main properties. Then, we show the current best-known solutions (BKS) for each instance.

Table VI.1 presents an overview of common instance sets for the 3L-CVRP and 3L-VRPTW. Hereby, the most relevant parameters are the number of customers (n) and the number of items (m). The most common instance set for the 3L-CVRP is created by Gendreau et al. (2006), which is extended by Tarantilis et al. (2009) with 12 more difficult instances. The 3L-VRPTW instance set by D. Zhang et al. (2017) is created by combining the two well-known instance sets provided by Gendreau et al. (2006) and Solomon (1987). Concerning the instances by Krebs et al. (2021), the instances vary systematically in the number of customers, items, and item types to enable detailed analysis of influencing parameters. Moreover, for all instance sets, axle weights were added based on realistic parameters. All instance sets are published at GitHub[2].

In terms of the instance set by Gendreau et al. (2006) shown in Table VI.2, the algorithm proposed in Z. Zhang et al. (2015) receives currently the best overall solutions indicated by the lowest average total travel distance. Therefore, most BKS are found by this algorithm. As detailed

[2]https://github.com/CorinnaKrebs/Instances

Table VI.1: Overview of Instance Sets

author	Problem	#	n	m
Gendreau et al. (2006)	3L-CVRP	27	[15, 100]	[26, 199]
Fuellerer et al. (2010)	3L-CVRP	12	[50, 125]	[73, 379]
D. Zhang et al. (2017)	3L-VRPTW	27	[15, 100]	[26, 199]
Krebs et al. (2021)	3L-VRPTW	600	20, 60, 100	200, 400

results for Z. Zhang et al. (2015) are available[3], we validated all results with the Solution Validator. However, some of the best solutions within the published data differ from the values described in Z. Zhang et al. (2015). Therefore, we report only the best-known results that were actually found in the data set. These validated and best-known results are published via GitHub[4]. The used algorithm shows its strength for instances with more than 32 customers (see instances 14-27). For instances with less customers, the algorithms by Escobar-Falcon et al. (2016), Tao and Wang (2015) and Bortfeldt (2012) find better solutions. However, for these solutions, validation was not possible due to summarized results.

Table VI.2: BKS for Gendreau et al. (2006) instances

no	*ttd*	reference	no	*ttd*	reference
1	300.69	Escobar-Falcon et al. (2016)	15	1338.22	Z. Zhang et al. (2015)
2	334.96	Z. Zhang et al. (2015)	16	698.61	Z. Zhang et al. (2015)
3	374.81	Escobar-Falcon et al. (2016)	17	866.40	Z. Zhang et al. (2015)
4	430.88	Escobar-Falcon et al. (2016)	18	1207.70	Bortfeldt (2012)
5	436.48	Tao and Wang (2015)	19	741.74	Bortfeldt (2012)
6	498.16	Bortfeldt (2012)	20	576.88	Z. Zhang et al. (2015)
7	767.46	Tao and Wang (2015)	21	1067.70	Z. Zhang et al. (2015)
8	804.75	Tao and Wang (2015)	22	1147.80	Bortfeldt (2012)
9	630.13	Z. Zhang et al. (2015)	23	1103.44	Z. Zhang et al. (2015)
10	820.35	Bortfeldt (2012)	24	1102.14	Z. Zhang et al. (2015)
11	772.85	Tao and Wang (2015)	25	1370.34	Z. Zhang et al. (2015)
12	610.23	Z. Zhang et al. (2015)	26	1557.15	Z. Zhang et al. (2015)
13	2608.68	Tao and Wang (2015)	27	1496.28	D. Zhang et al. (2017)
14	1368.40	Bortfeldt (2012)	**avg**	**927.15**	

In Table VI.3, the BKS for the instance set by Tarantilis et al. (2009) are presented. As for the other 3L-CVRP instance set, most of the BKS are found by Z. Zhang et al. (2015). As detailed results are available, these solutions are checked with the Solution Validator and are published via GitHub[4], except for instance no 34, where a feasible solution could not be found.

Table VI.3: BKS for Tarantilis et al. (2009) instances

no	*ttd*	reference	no	*ttd*	reference
28	1417.88	Z. Zhang et al. (2015)	34	2595.22	Bortfeldt (2012)
29	2189.27	Z. Zhang et al. (2015)	35	4163.02	Z. Zhang et al. (2015)
30	1713.82	Z. Zhang et al. (2015)	36	3400	Z. Zhang et al. (2015)
31	2010.58	Z. Zhang et al. (2015)	37	3159.15	Z. Zhang et al. (2015)
32	2971.58	Z. Zhang et al. (2015)	38	5315.97	Z. Zhang et al. (2015)
33	2339.3	Z. Zhang et al. (2015)	39	4031.62	Z. Zhang et al. (2015)
			avg	**2942.28**	

[3] see https://alim.algorithmexchange.com/orlib/topic/3L-FCVRP/;jsessionid=BE16D2007BD713BBFCCE3A5926C6EFC0#Zhang2015

[4] https://github.com/CorinnaKrebs/BestKnownResults

Concerning the 3L-VRPTW instances by D. Zhang et al. (2017), the BKS are presented in Table VI.4. All solutions are found by the algorithms described in Krebs et al. (2023). As before, the validated results are published via Github[4].

Table VI.4: BKS for D. Zhang et al. (2017) Instances

Name	v_{used}	*ttd*	*time*	**Name**	v_{used}	*ttd*	*time*
VRPTWP01	4	245.44	3.95	VRPTWP15	8	527.62	98.78
VRPTWP02	5	276.64	1.39	VRPTWP16	11	693.92	5.72
VRPTWP03	4	274.55	30.01	VRPTWP17	14	951.11	17.31
VRPTWP04	6	336.79	2.39	VRPTWP18	12	979.93	33.89
VRPTWP05	6	345.89	17.81	VRPTWP19	12	971.43	127.27
VRPTWP06	6	374.22	2.83	VRPTWP20	17	1,311.32	765.60
VRPTWP07	5	324.29	22.06	VRPTWP21	16	1,189.80	1,943.54
VRPTWP08	6	320.75	21.19	VRPTWP22	18	1,466.27	305.06
VRPTWP09	9	458.32	19.40	VRPTWP23	17	1,325.73	690.63
VRPTWP10	7	487.60	58.50	VRPTWP24	16	1,289.15	708.50
VRPTWP11	7	493.58	114.46	VRPTWP25	20	1,432.66	3,600.00
VRPTWP12	9	575.04	14.76	VRPTWP26	24	1,642.74	1,455.63
VRPTWP13	6	452.05	467.32	VRPTWP27	22	1,597.13	2,111.58
VRPTWP14	8	550.16	89.08				
				Total	**295**	**20,894.11**	**12,728.66**

Table VI.5 presents the best-known results for the instance set by Krebs et al. (2021). As before, all solutions are found by the hybrid algorithms presented in Krebs et al. (2023). The detailed results are published via GitHub[4].

Table VI.5: BKS for Krebs et al. (2021) Instances

n	**m**	**types**	**sum** v_{used}	**sum** *ttd*	**avg** *time*
20	200	3	73	8,163.90	767.16
		10	71	8,351.98	1,804.59
		100	73	8,444.32	1,817.26
	400	3	88	8,834.64	2,340.33
		10	95	9,215.81	2,673.61
		100	106	9,694.34	3,172.97
60	200	3	430	39,940.64	1,563.98
		10	448	40,562.89	2,027.61
		100	464	41,765.02	2,163.44
	400	3	693	53,718.93	2,287.25
		10	719	55,179.40	2,563.58
		100	782	59,381.69	2,964.56
100	200	3	465	47,770.81	2,024.01
		10	510	50,906.28	2,330.88
		100	548	54,124.17	2,146.65
	400	3	797	66,799.70	2,702.32
		10	865	71,842.97	2,937.54
		100	878	72,187.87	2,967.32
Total			**8,105**	**706,885.36**	**2,331.14**

6 Summary and Future Work

In this paper, two open-source tools are presented for the combined Vehicle Routing and Container Loading Problem (alias “3L-CVRP" and “3L-VRPTW"). The Solution Validator checks the feasibility of solutions in terms of considered constraints. The Visualizer displays the solutions in separated views. Both tools are also suitable for the usage of each optimization problem.

In the paper, all necessary data, the access, the requirements, and the usage of the tools are demonstrated. Using these tools can be beneficial for further research: Through the Solution Validator, the feasibility of solutions can be ensured and the results can be published online to increase transparency. Moreover, solutions can be checked concerning different loading constraints and various formulations which gives insights into the restrictiveness and usage of loading constraints. The Visualizer provides information about the solution and visualizes the entire routing and loading process step by step. Further analysis can reveal weaknesses and therefore lead to improvements in the solution approaches. The tools are fully adaptable as the source code is well documented and published online. As future work, the tools are improved by including new features or removing currently unknown bugs.

Conflict of interest

The authors declare that they have no conflict of interest.

References

Baker, B., Coffman, E., and Rivest, R. (1980). "Orthogonal Packings in Two Dimensions". In: *SIAM Journal on Computing* vol. 9, no. 4, pp. 846–855. DOI: 10.1137/0209064.

Bischoff, E. E. (2003). "Dealing with Load Bearing Strength Considerations in Container Loading Problems". In.

Bischoff, E. E. and Ratcliff, M. S. W. (1995). "Issues in the development of approaches to container loading". In: *Omega* vol. 23, no. 4, pp. 377–390. ISSN: 03050483. DOI: 10.1016/0305-0483(95)00015-G.

Bortfeldt, A. (2012). "A hybrid algorithm for the capacitated vehicle routing problem with three-dimensional loading constraints". In: *Computers & Operations Research* vol. 39, no. 9, pp. 2248–2257. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2011.11.008.

Bortfeldt, A. and Yi, J. (2020). "The Split Delivery Vehicle Routing Problem with three-dimensional loading constraints". In: *European Journal of Operational Research* vol. 282, no. 2, pp. 545–558. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2019.09.024. URL: https://www.sciencedirect.com/science/article/pii/S0377221719307647.

Ceschia, S., Schaerf, A., and Stützle, T. (2013). "Local search techniques for a routing-packing problem". In: *Computers and Industrial Engineering* vol. 66, no. 4, pp. 1138–1149. ISSN: 0360-8352. DOI: https://doi.org/10.1016/j.cie.2013.07.025. URL: http://www.sciencedirect.com/science/article/pii/S0360835213002404.

Escobar-Falcon, L. M., Ãlvarez-Martinez, D., Granada-Echeverri, M., Escobar, J. W., and Romero-LÃ¡zaro, R. A. (Mar. 2016). "A matheuristic algorithm for the three-dimensional loading capacitated vehicle routing problem (3L-CVRP)". en. In: *Revista Facultad de IngenierÃa Universidad de Antioquia*, pp. 09–20. ISSN: 0120-6230. URL: http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S0120-62302016000100002&nrm=iso.

Fuellerer, G., Doerner, K. F., Hartl, R. F., and Iori, M. (2010). "Metaheuristics for vehicle routing problems with three-dimensional loading constraints". In: *European Journal of Operational Research* vol. 201, no. 3, pp. 751–759. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2009.03.046. URL: http://www.sciencedirect.com/science/article/pii/S0377221709002252.

Gendreau, M., Iori, M., Laporte, G., and Martello, S. (2006). "A Tabu Search Algorithm for a Routing and Container Loading Problem". In: *Transportation Science* vol. 40, no. 3, pp. 342–350. ISSN: 0041-1655. DOI: 10.1287/trsc.1050.0145. URL: http://pubsmisc.informs.org/doi/abs/10.1287/trsc.1050.0145.

Iori, M., Salazar González, J. J., and Vigo, D. (May 2007). "An Exact Approach for the Vehicle Routing Problem with Two-Dimensional Loading Constraints". In: *Transportation Science* vol. 41, pp. 253–264. DOI: 10.1287/trsc.1060.0165.

Junqueira, L., Oliveira, J. F., Carravilla, M. A., and Morabito, R. (2013). "An optimization model for the vehicle routing problem with practical three-dimensional loading constraints". In: *International Transactions in Operational Research* vol. 20, no. 5, pp. 645–666. DOI: https://doi.org/10.1111/j.1475-3995.2012.00872.x. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.

1111/j.1475-3995.2012.00872.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1475-3995.2012.00872.x.

Koch, H., Bortfeldt, A., and Wäscher, G. (Feb. 2018). "A hybrid algorithm for the vehicle routing problem with backhauls, time windows and three-dimensional loading constraints". In: *OR Spectrum* vol. 40. DOI: 10.1007/s00291-018-0506-6.

Krebs, C. and Ehmke, J. F. (2021a). "Axle Weights in combined Vehicle Routing and Container Loading Problems". In: *EURO Journal on Transportation and Logistics* vol. 10, p. 100043. ISSN: 2192-4376. DOI: 10.1016/j.ejtl.2021.100043. URL: https://www.sciencedirect.com/science/article/pii/S2192437621000157.

— (2021b). "Vertical Stability Constraints in Combined Vehicle Routing and 3D Container Loading Problems". In: *Computational Logistics*. Ed. by Mes, M., Lalla-Ruiz, E., and Voß, S. Cham: Springer International Publishing, pp. 442–455. ISBN: 978-3-030-87672-2.

Krebs, C., Ehmke, J. F., and Koch, H. (Aug. 2021). "Advanced loading constraints for 3D vehicle routing problems". In: *OR Spectrum*. ISSN: 1436-6304. DOI: 10.1007/s00291-021-00645-w. URL: https://doi.org/10.1007/s00291-021-00645-w.

— (2023). "Effective loading in combined vehicle routing and container loading problems". In: *Computers & Operations Research* vol. 149, p. 105988. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2022.105988. URL: https://www.sciencedirect.com/science/article/pii/S0305054822002258.

Lodi, A., Martello, S., and Vigo, D. (Nov. 1999). "Heuristic and Metaheuristic Approaches for a Class of Two-Dimensional Bin Packing Problems". In: *INFORMS Journal on Computing* vol. 11, no. 4, pp. 345–357. DOI: 10.1287/ijoc.11.4.345. URL: https://ideas.repec.org/a/inm/orijoc/v11y1999i4p345-357.html.

Mack, D., Bortfeldt, A., and Gehring, H. (2004). "A parallel hybrid local search algorithm for the container loading problem". In: *International Transactions in Operational Research* vol. 11, no. 5, pp. 511–533. DOI: 10.1111/j.1475-3995.2004.00474.x.

Moura, A. (2008). "A Multi-Objective Genetic Algorithm for the Vehicle Routing with Time Windows and Loading Problem". In: *Intelligent Decision Support: Current Challenges and Approaches*. Ed. by Bortfeldt, A., Homberger, J., Kopfer, H., Pankratz, G., and Strangmeier, R. Wiesbaden: Gabler, pp. 187–201. ISBN: 978-3-8349-9777-7. DOI: 10.1007/978-3-8349-9777-7. URL: https://doi.org/10.1007/978-3-8349-9777-7.

Moura, A. and Oliveira, J. F. (Oct. 2009). "An integrated approach to the vehicle routing and container loading problems". In: *OR Spectrum* vol. 31, no. 4, pp. 775–800. ISSN: 1436-6304. DOI: 10.1007/s00291-008-0129-4. URL: https://doi.org/10.1007/s00291-008-0129-4.

Pace, S., Turky, A., Moser, I., and Aleti, A. (2015). "Distributing Fibre Boards: A Practical Application of the Heterogeneous Fleet Vehicle Routing Problem with Time Windows and Three-dimensional Loading Constraints". In: *Procedia Computer Science* vol. 51. International Conference On Computational Science, ICCS 2015, pp. 2257–2266. ISSN: 1877-0509. DOI: 10.1016/j.procs.2015.05.382. URL: http://www.sciencedirect.com/science/article/pii/S1877050915011904.

Potel, M. (1996). "MVP: Model-View-Presenter The Taligent Programming Model for C++ and Java". In.

Ramos, A., Silva, E., and Oliveira, J. (Oct. 2017). "A new Load Balance Methodology for Container Loading Problem in Road Transportation". In: *European Journal of Operational Research* vol. 266. DOI: 10.1016/j.ejor.2017.10.050.

Solomon, M. M. (1987). "Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints". In: *Operations Research* vol. 35, no. 2, pp. 254–265. DOI: 10.1287/opre.35.2.254.

Tao, Y. and Wang, F. (2015). "An effective tabu search approach with improved loading algorithms for the 3L-CVRP". In: *Computers & Operations Research* vol. 55, pp. 127–140. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2013.10.017. URL: https://www.sciencedirect.com/science/article/pii/S0305054813003122.

Tarantilis, C. D., Zachariadis, E. E., and Kiranoudis, C. T. (June 2009). "A Hybrid Metaheuristic Algorithm for the Integrated Vehicle Routing and Three-Dimensional Container-Loading Problem". In: *IEEE Transactions on Intelligent Transportation Systems* vol. 10, no. 2, pp. 255–271. ISSN: 1524-9050. DOI: 10.1109/TITS.2009.2020187.

Zhang, D., Cai, S., Ye, F., Si, Y.-W., and Nguyen, T. T. (2017). “A hybrid algorithm for a vehicle routing problem with realistic constraints”. In: *Information Sciences* vol. 394-395, pp. 167–182. ISSN: 0020-0255. DOI: 10.1016/j.ins.2017.02.028.

Zhang, Z., Wei, L., and Lim, A. (2015). “An evolutionary local search for the capacitated vehicle routing problem minimizing fuel consumption under three-dimensional loading constraints”. In: *Transportation Research Part B: Methodological* vol. 82, pp. 20–35. ISSN: 0191-2615. DOI: https://doi.org/10.1016/j.trb.2015.10.001.

Appendix

```
Name                          3l_cvrp01
Number_of_Customers           15
Number_of_Items               32
Number_of_ItemTypes           32
Number_of_Vehicles            4
TimeWindows                   0

VEHICLE
Mass_Capacity                 90
CargoSpace_Length             60
CargoSpace_Width              25
CargoSpace_Height             30
Wheelbase                     48
Max_Mass_FrontAxle            50
Max_Mass_RearAxle             82
Distance_FrontAxle_CargoSpace 4

CUSTOMERS
i             x           y           Demand        ReadyTime     DueDate       ServiceTime   DemandedMass  DemandedVolume
0             30          40          0             0             0             0             0             0
1             37          52          1             0             0             0             7             1050
...
15            36          16          3             0             0             0             10            11448

ITEMS
Type          Length      Width       Height        Mass          Fragility     LoadBearingStrength
Bt1           30          5           7             7             1             0.9188947
...
Bt32          34          6           9             3.33          1             0.9580698

DEMANDS PER CUSTOMER
i       Type Quantity
1       Bt1 1
...
15      Bt30 1  Bt31 1  Bt32 1
```

Figure VI.6: Exemplary Instance File

```
// Parameter for the Constraints
alpha           0.75            // Support Parameter                            in %
lambda          5               // distance for reachability                    in dm
balanced_part   0.7             // Part of balanced loading                     in %

// (De-)activation of Constraints
rotation                1       // Rotation of Items            (0: n, 1: y)
capacity                1       // Load Capacity                (0: n, 1: y)
unloading_sequence      1       // Unloading Sequence           (0: n; 1: lifo,         2: mlifo)
vertical_stability      1       // Vertical Stability           (0: n, 1: minimal support area,
                                                                    2: robust stability (multiple overhanging),
                                                                    3: robust stability 2 (top overhanging))
stacking                1       // Stacking Constraints         (0: n, 1: fragility,    2: LBS Simplified,      3: LBS Complete)
reachability            0       // Reachability                 (0: n, 1: y)
axle_weights            0       // Axle Weight Constraint       (0: n, 1: y)
balancing               0       // Balanced Loading             (0: n, 1: y)
```

Figure VI.7: Constraint File with Basic Constraint Set

```
Name:                           31_cvrp01
Problem:                        3L-CVRP
Number_of_used_Vehicles:        4
Total_Travel_Distance:          302.02
Calculation_Time:               3.39
Total_Iterations:               12277
ConstraintSet:                  P1

-------------------------------------------------------------------------------------------------------------------
Tour_Id:                        1
No_of_Customers:                5
No_of_Items:                    8
Customer_Sequence:              11 2 9 10 5

CustId  Id      TypeId  Rotated x       y       z       Length  Width   Height  mass    Fragility       LoadBearingStrength
10      17      17      0       0       0       0       25      12      14      5       0               2.076255
...
11      20      20      0       44      0       15      16      13      10      6.33    0               3.325301


-------------------------------------------------------------------------------------------------------------------
Tour_Id:                        2
```

Figure VI.8: Exemplary Solution File

Chapter 2

Conclusion

This thesis focuses on new loading constraints' design and introduction, the improvement of existing formulations, and new approaches for the combined vehicle and 3D container loading problem introduced by Gendreau et al. 2006, 3L-CVRP, and the extension 3L-VRPTW. Both optimization problems deal with a number of customers, each with a demand represented by a number of 3D parcels lying at the depot. A fleet of heterogeneous vehicles must fulfill this demand. Moreover, various routing and loading constraints must be obeyed. The extension 3L-VRPTW considers additional time windows in which the delivery must take place.

In the basic version of the problem, six loading constraints (*basic set*) are obeyed: Geometry, Orthogonality, Rotation, LIFO, Minimal Supporting Area, and Fragility. Although dozens of papers deal with the 3L-CVRP and variants, few deviate from the *basic set*. The loading constraints are embedded in a hybrid algorithm, which is necessary to solve the combined problem. Therefore, the hybrid algorithm by Koch et al. 2018 is used. It consists of a modified Adaptive Large Neighborhood Search originally introduced by Ropke and Pisinger 2006 for solving the routing part. The loading part is solved using a Deepest-Bottom-Left-Fill algorithm initially implemented by Karabulut and İnceoğlu 2005. The idea is to place items as far rearwards, bottom-wards, and leftwards inside of the vehicle loading space as possible. The available placement positions for the items are stored as points. Whenever a placement position for an item is selected, the position is checked regarding its feasibility according to loading constraints.

This thesis's first achievement is its improved implementation of the Deepest-Bottom-Left-Fill variant: The available placement positions are represented by free spaces, reducing the complexity of checking items that overlap with other items or vehicle walls (Geometry constraint). Moreover, the items are slid to fill free spaces. As a result, the total travel distance decreases by around 3% and the runtime by around 5%, on average, compared to the original algorithm by Koch et al. 2018.

Moreover, computational experiments are conducted for well-known loading constraints from literature, and new formulated or improved loading constraints are introduced in the papers. In the following, all loading constraints are briefly described and their impact on the objective value and the runtime is presented. To enable a fair comparison, the computational results are evaluated based on one common objective function, i.e. minimizing the total travel distance and expressed as a deviation to the *basic set*. The average of all computational results is used which are received for the instance set by Krebs et al. 2021. The impact for loading constraints, which are included in the *basic set*, i.e. Rotation, Minimal Supporting Area, Fragility, and LIFO, is measured by the reversed impact when excluding from the model. In terms of the "Reloading Effort" constraint, the instance set is not applicable because not all items of the instance set can be handled manually. Therefore, its impact is based on a specific instance set and these values are shown.

- *Rotation*
  This constraint allows for items to be rotated along the length-width plane. As the computational experiments show, the increase of the solution space through the possibility of rotating items is also reflected in the objective function. If the rotation of items is not allowed, the total travel distance increases by 1.76% on average, while the runtime decreases by almost 30%. The usage of this constraint depends on the overall target: If solutions should be received fast, then the rotation constraint should not be included. If the objective value is more relevant, the constraint has a positive impact.

- *Unloading Sequence*
  This category deals with the unloading sequence of the items and specific requirements. The

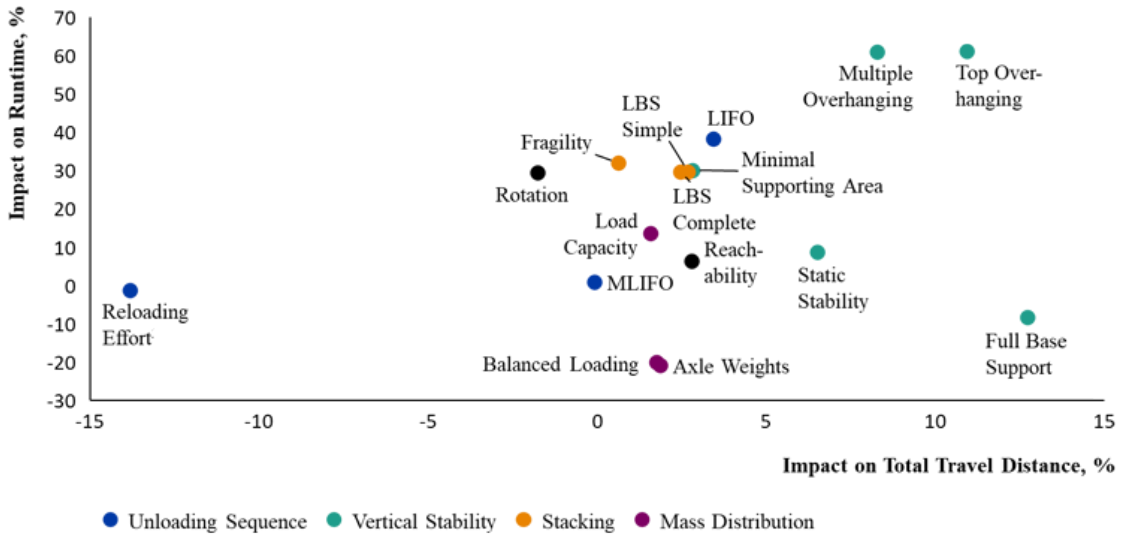


Figure 2.1: Comparison of Loading Constraints

purpose is to ensure the smooth loading and unloading of items. The recommendation of which constraint to choose depends on the dimensions of the items: If items can be handled manually, the constraint "Reloading Effort" instead of the LIFO constraint should be used as it achieves the best results. If equipment for unloading is required, the LIFO constraint is the better choice as it enables higher flexibility for the equipment compared to MLIFO.

  - *LIFO*
    The items are unloaded through straight movements toward the back door. Thus, items that are delivered later are not allowed to hang over those that are delivered earlier. The results show that, when disregarding the LIFO constraint, the total travel distance improves for half of the instances. On average, the total travel distance decreases by 3.43%. In the case of the runtime, it decreases on average by almost 40%.
  - *Manual LIFO*
    As in LIFO, the items are unloaded through straight movements. In contrast to the LIFO constraint, earlier-delivered items can hang over those delivered later. The computational experiments show that the Manual LIFO constraint has merely no effect on the total travel distance (on average: -0.09%). The same applies to the runtime with even an average increase of 0.93%.
  - *Reloading Effort*[1]
    This approach deals with the reloading effort of items when loading or unloading. This enables that items can block other items during unloading. Therefore, it can replace the LIFO constraint. This time-wise effort is calculated using the well-known "Methods Time Measurement" (MTM) and is added to the objective value. The constraint leads to a decrease of 13.82% in the total travel distance and -1.25% in the runtime, on average, based on a specific newly-introduced instance set.

- *Vertical Stability*
  These constraints deal with vertical stability to prevent objects from falling on the ground, on top of other items, or even on the driver. The vertical stability constraints thus significantly impact the total travel distance and the runtime. Here, the recommendation is to use the "Static Stability" constraint as it ensures high stability of item stacks with less impact on the total

[1]Introduced in this thesis.

travel distance and runtime compared to constraints with higher stability guarantee (i.e. ”Full Base Support” or ”Top Overhanging”).

  - *Minimal Supporting Area*
    This formulation requires that the base area of a stacked item is covered by a certain percentage α by other direct underlying items. However, as further analysis shows, the constraint does not ensure feasible stacks of items. To measure the impact on the total travel distance based on the *basic set*, it must be excluded from the model for the computational experiments. Thus, if there is no vertical stability requirement, the total travel distance improves by 2.81% and the runtime by 30% on average.
  - *Full Base Support*
    In contrast to other vertical stability formulations, the full base support prevents items from hanging over others. As this formulation has the highest restrictiveness among vertical stability constraints, the total travel distance shows the highest increases with 12.73%, on average. However, the runtime decreases by 8.3%.
  - *Multiple Overhanging*
    This constraint is based on the Minimal Supporting Area constraint and is aimed to improve the formulation to ensure stable stacks. It states that the Minimal Supporting Area is obeyed at any height. This formulation has a significant impact on the total travel distance. Almost every instance shows an increase in the total travel distance. On average, it is 8.27%; the runtime increases by 61.01%, on average.
  - *Top Overhanging*[1]
    The thesis introduces the Top Overhanging constraint. In this concept, all items must be fully supported by direct underlying items except the last one of a stack. The idea is to reduce the computational complexity compared to the Multiple Overhanging constraint and to ensure stable stacks in contrast to the Minimal Supporting Area constraint. However, due to the restrictiveness of this constraint, the total travel distance increases by 10.93% on average, and the runtime by 61.33%.
  - *Static Stability*[1]
    Based on existing corner cases and the science of statics, another constraint in the vertical stability category has been developed in this thesis. Thus, the Minimum Supporting Area must be obeyed, and an item's center of gravity must lie inside all levels at any height. Each level is determined by the minimum and maximum edges of items at the same height. As the computational experiments show, the Static Stability constraint performs the best regarding the runtime and the objective values compared to other constraints, ensuring stable stacking in Vertical Stability: On average, the total travel distance increases by 6.51% and the runtime by 8.72%.

- *Stacking*
  This category ensures that one item is not damaged when placed under another. The stacking constraints have rather a high impact on the total travel distance and the runtime. Since the Load Bearing Strength – Complete constraint best reflects real stacking situations and load distributions, the recommendation is to use this constraint rather than other stacking constraints.

  - *Fragility*
    In this constraint, the items are classified as fragile and non-fragile items. Non-fragile items can only be placed on top of other non-fragile items to prevent items from being smashed by other items. If the constraint is excluded from the constraint set, the total travel distance rather does not improve (on average: -0.63%). In contrast, the total travel distance increases by 2% despite the increased solution space. At least, the runtime decreases by 32.14% on average.

- *Load Bearing Strength – Simplified*
  As more information regarding the items becomes available, detailed data on the masses of the items and the maximum bearing load exist. The idea of load-bearing strength constraints is to use this data to calculate the acting load on the surface of an item caused by stacking. The simplified approach distributes the acting load to all items underneath an item's base area. The total travel distance increases by 2.69% on average, while the constraint has a high impact on runtime, increasing it by 29.71%.
- *Load Bearing Strength – Complete*[1]
  Compared to the previous constraint (the Load Bearing Strength – Simplified), this constraint distributes loads based on the science of statics, as introduced in this thesis. First, the load acting on an item is distributed to the directly underlying items. Then, their load is further distributed until the items are on the ground. The computational results are rather similar to the other approach: The total travel distance increases by 2.47% and the runtime by 29.75% on average. However, as this approach is more realistic, this should be used instead of the simplified one.

- *Mass Distribution*
  These constraints influence the items' position within the vehicle's loading space, emphasizing mass distribution. The constraints have relatively small negative effects on the total travel distance but high positive effects on the runtime. In terms of Axle Weights and Balanced Load constraints, they also improve the security of vehicles. Therefore, all of these constraints should be included in the model.

  - *Load Capacity*
    Normally, each item has a defined mass and the vehicle has a maximum load capacity D. Computational experiments show that when excluding the load capacity (i.e., setting the item's masses to zero) the total travel distance improves only slightly (-1.57%), while the runtime shows higher positive effects (-13.69%). As the Load Capacity constraint is required for the following constraints (Axle Weighs and Balanced Load) and is a widely-used constraint, it should be kept in the model.
  - *Axle Weights*[1]
    In this thesis, formulas for the calculation of axle weights for various types of vehicles are presented. These formulas are based on the science of statics and can be adapted to trucks with or without trailers and different axle configurations. The consideration of this constraint shows relatively small effects on the total travel distance with an increase of 1.87%, on average. However, the runtime even decreases by 20.83%, on average. Since axle weights are a legal requirement, the slight increase of the objective value, and the positive effect on the runtime, the recommendation is to include this constraint in the model.
  - *Balanced Load*
    This constraint requires that the sum of item masses per vehicle half does not exceed a certain percentage of the load capacity of a vehicle. It aims at preventing the vehicle from tipping over due to unbalanced loads. The constraint shows the same tendencies as the axle weight constraint: It has relatively minor effects on the total travel distance with only 1.76%, on average. Moreover, the runtime decreases by 20.04%, on average. As this constraint has positive effects on the runtime, negligible effects on the objective value, and increases the security of the vehicle, items, and road users, this constraint should always be included.

- *Reachability*
  Items might be blocked by or on top of different items, making them unreachable. To avoid these situations, the constraint guarantees that the item is reachable according to a certain length. The evaluation of the computational experiments shows that the constraint has a small

impact on the total travel distance with an average increase of 2.8% while the runtime increases by 6.38%, on averge. Because this constraint guarantees security for the items and the driver, who might need to stand on top of other items, the recommendation is to include this constraint in the model.

To summarize, the different loading constraints highly differ in the impact on the objective value and the runtime. Vertical stability constraints have the most significant effect, followed by stacking constraints (Fragility and Load Bearing Strength) and the Unloading Sequence (LIFO). Previous evaluations focused on the impact of one loading constraint. Therefore, another analysis deals with the effect of the current largest constraint set (Rotation, LIFO, Top Overhanging, Load Bearing Strength – Complete, Reachability, Axle Weights, and Balanced Load). Its results show that the total travel distance increases (on average 17.15%) and the runtime by 63.08%, but not as much as the sum of the individual constraints.

As the last part of the thesis, two tools have been created to support further research. The first tool, "Solution Validator," checks provided solutions for their correctness and feasibility according to routing and loading constraints. For this purpose, all constraints are implemented and well documented. The tool outputs a report indicating the feasibility, and in case of infeasibility, it presents violated constraints together with the corresponding vehicle, customer, or item. The second tool, "Visualizer," visualizes the solution through two views: One for the routing part showing the tours and the corresponding schedule with driving, waiting, and service times, with the second representing the loading plan with all the items and their positions within the vehicle loading space. All necessary data regarding the items and their feasibility are presented in a table. Both the tour and loading processes can be performed step by step.

Outlook

There are several aspects for future research. First, the recommendation is to develop specialized algorithms adapted to the properties of items and the most critical loading constraint. For example, for items with very high density, an algorithm focusing on the observance of the axle loads and the balanced load would be an interesting pathway for future research.
Second, relaxing routing constraints enables new possibilities for new research topics. When allowing multiple visits per customer (split delivery), the impact on the objective values can be analyzed. Furthermore, new loading constraints are possible in the context of split delivery, e.g., the necessity that specific items cannot be loaded in the same vehicle loading space. This constraint is applied when transporting chemicals or conflicting animals.
To conclude, this thesis covers numerous real-world loading constraints. Nevertheless, several further modifications and extensions have not yet been considered, providing intriguing research questions for future projects.

References

Gendreau, M., Iori, M., Laporte, G., and Martello, S. (2006). "A Tabu Search Algorithm for a Routing and Container Loading Problem". In: *Transportation Science* vol. 40, no. 3, pp. 342–350. ISSN: 0041-1655. DOI: 10.1287/trsc.1050.0145. URL: http://pubsmisc.informs.org/doi/abs/10.1287/trsc.1050.0145.

Karabulut, K. and İnceoğlu, M. M. (2005). "A Hybrid Genetic Algorithm for Packing in 3D with Deepest Bottom Left with Fill Method". In: *Advances in Information Systems*. Ed. by Yakhno, T. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 441–450. ISBN: 978-3-540-30198-1.

Koch, H., Bortfeldt, A., and Wäscher, G. (Feb. 2018). "A hybrid algorithm for the vehicle routing problem with backhauls, time windows and three-dimensional loading constraints". In: *OR Spectrum* vol. 40. DOI: 10.1007/s00291-018-0506-6.

Krebs, C., Ehmke, J. F., and Koch, H. (Aug. 2021). "Advanced loading constraints for 3D vehicle routing problems". In: *OR Spectrum*. ISSN: 1436-6304. DOI: 10.1007/s00291-021-00645-w. URL: https://doi.org/10.1007/s00291-021-00645-w.

Ropke, S. and Pisinger, D. (2006). "A unified heuristic for a large class of Vehicle Routing Problems with Backhauls". In: *European Journal of Operational Research* vol. 171, no. 3. Feature Cluster: Heuristic and Stochastic Methods in Optimization Feature Cluster: New Opportunities for Operations Research, pp. 750–775. ISSN: 0377-2217. DOI: 10.1016/j.ejor.2004.09.004. URL: http://www.sciencedirect.com/science/article/pii/S0377221704005831.

The logistics world is undergoing a digital transformation, opening new opportunities for data-driven and realistic logistics planning. In this thesis, the focus on highly constrained loading problems within container loading and vehicle routing problems leads to the introduction of new complex loading constraints and the refinement of existing ones. By examining the associated costs and benefits, the impact of each loading constraint is evaluated. Comprising six scientific papers, this research covers various aspects of the field: From algorithmic improvements to novel formulations, each paper sheds light on the optimization puzzle from a unique perspective: Discover the power of the Deepest-Bottom-Left-Fill algorithm, dive into a comprehensive analysis of complex loading constraints, understand the importance of axle weight considerations, witness the improvement of load bearing strength constraints, explore realistic formulations for vertical stability, and experience adaptations for manual loading with time-wise integration. Finally, tools for validation and visualization are introduced, which provide a comprehensive view on solutions. This intriguing exploration unlocks the potential for efficient logistics operations, merging digital transformation with optimized loading constraints for a future of efficiency.