



Solution validator and visualizer for (combined) vehicle routing and container loading problems

Corinna Krebs¹ · Jan Fabian Ehmke²

Accepted: 2 February 2023
© The Author(s) 2023

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.

Keywords Open source tools · Vehicle routing problem · Container loading problem

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

✉ Corinna Krebs
Corinna.Krebs@ovgu.de

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

¹ Management Science, Otto von Guericke University Magdeburg, Universitätsplatz 2, 39106 Magdeburg, Germany

² Business Decisions and Analytics, University of Vienna, Kolingasse 14-16, 1090 Vienna, Austria

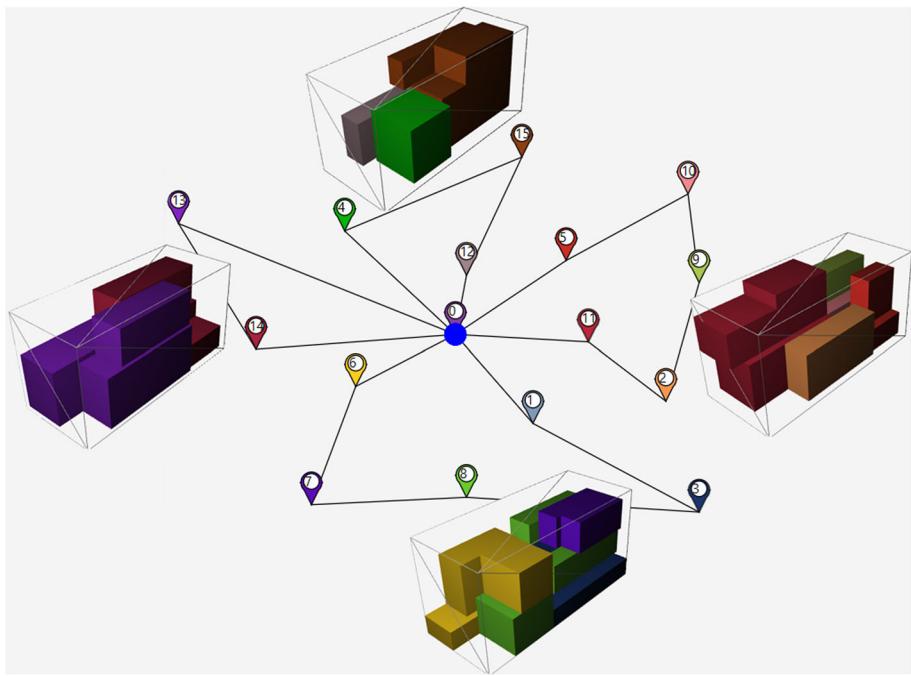


Fig. 1 Exemplary solution for instance “3l-cvrp01”

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. 1.

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 & 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

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.

“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 Sect. 2. To introduce the 3L-CVRP and show the covered features, the problems are formulated in Sect. 3. In Sect. 4 both tools are introduced and their application is explained. In Sect. 5, we present an overview of available instance sets, their properties, and the best-known solutions in the literature. Finally, Sect. 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.

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 et al., 2012; Escobar et al., 2016; 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. 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 $e_{i,j} \in E$ ($i \neq j, i, j = 0, \dots, n$) 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 $I_{i,k}$ ($k = 1, \dots, c_i$) 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 ($v = 1, \dots, v_{used}$). 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 Sect. 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. 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 Sect. 3.3). A complete *Constraint File* defining the *basic constraint set* by Gendreau et

al. (2006) is presented in Fig. 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. 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 Sect. 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 Sect. 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.

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

Code 1 Original Java Solution Validator Program

```

1  public static void main(String[] args) {
2      Instance instance = Read.readInstanceFile(pathToInstanceFile);
3      ConstraintSet constraintSet =
4          Read.readConstraintFile(pathToConstraintFile);
5      Solution solution = Read.readSolutionFile(pathToSolutionFile,
6          instance);
6      if (checkRoutingConstraints(solution, constraintSet, instance,
7          true))
7          && checkLoadingConstraints(solution, constraintSet, instance,
8          true)) {
9          System.out.println("All Constraints checked. Solution is
10         feasible.");
11         System.exit(1);
12     }
13     System.err.println("Solution is not feasible. Please check error
14         hints above.");
15     System.exit(-1);
16 }
```

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      ...
6 }
```

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, Sect. 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.

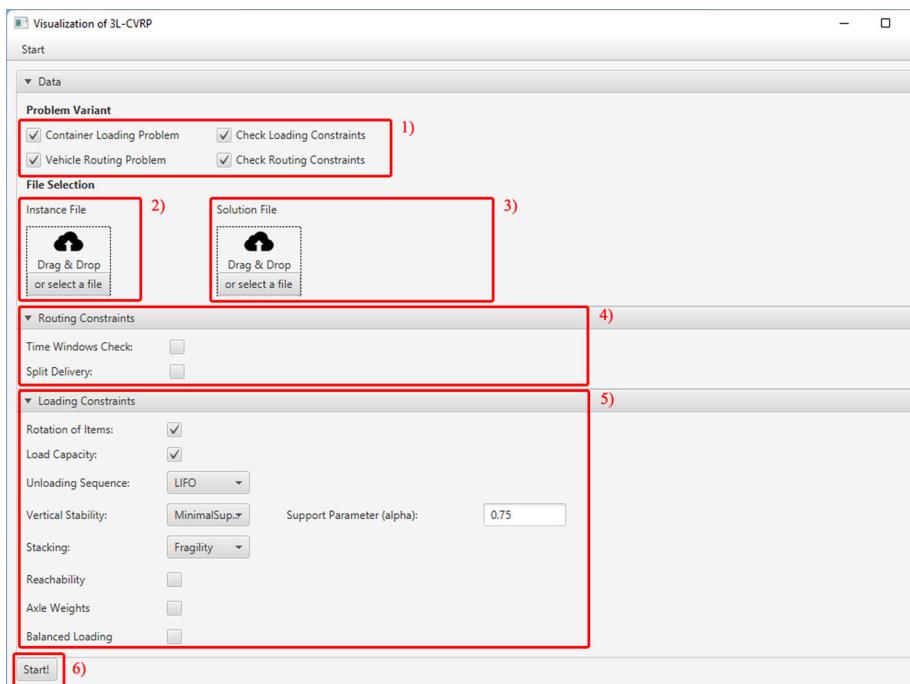


Fig. 2 View of the data input mask

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. 2).

In the following, the areas shown in Fig. 2 are described in more detail:

- 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.
- 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.
- 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.
- 2.4 In the center area of this view, additional routing constraints for the feasibility check can be included.
- 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 Sect. 3, a drop-down list for the selection appears.

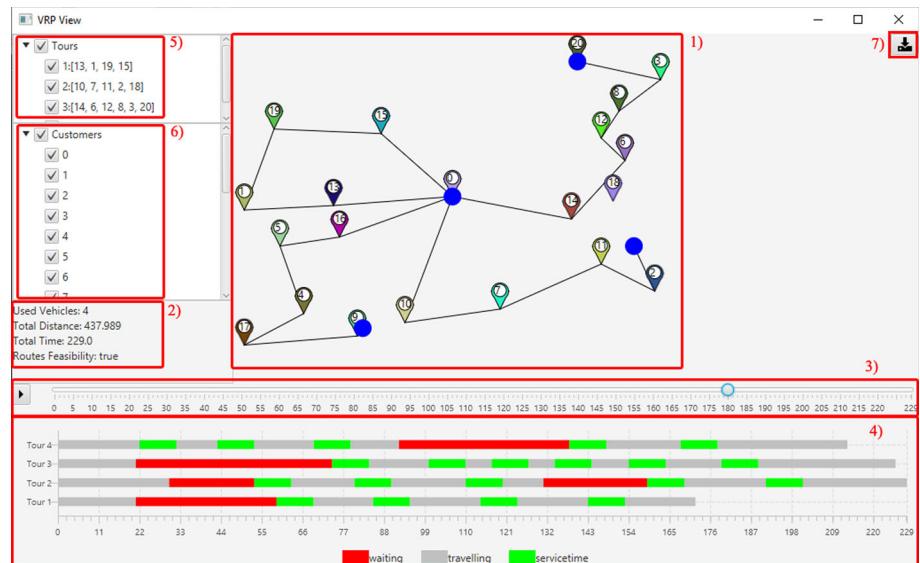


Fig. 3 View of the vehicle routing problem

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. 3. Directly after opening, an animation starts showing the complete routing process.

Regarding Fig. 3, the areas describe the following:

- 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.
- 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.
- 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.
- 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.
- 3.5 This tree enables changing the visibility of tours. This might be helpful in case of hidden details.
- 3.6 As in the previous item, this tree changes the visibility of the customer pins.

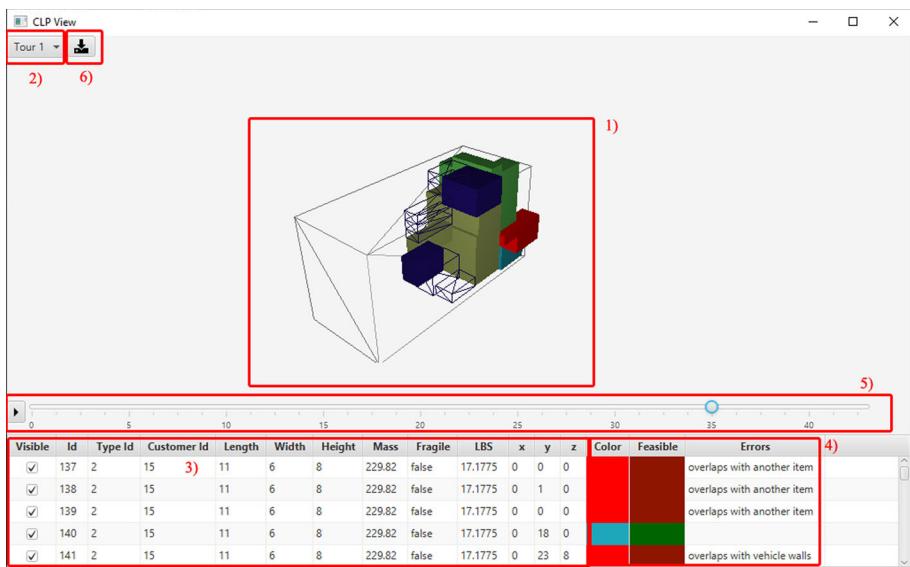


Fig. 4 View of the container loading problem

3.7 Through the download button, it is possible to download the currently displayed area for further research purposes.

In Fig. 4, the CLP view is presented. Similar to the VRP view, an animation showing the loading process starts directly after opening the view.

As shown in Fig. 4, there are six areas in the CLP view:

- 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.
- 4.2 Through the drop-down menu, it is possible to switch the tours and consequently, to change the loading space in the center.
- 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.
- 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.
- 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.
- 4.6 The download button enables downloading a picture of the displayed loading space with its current loading step.

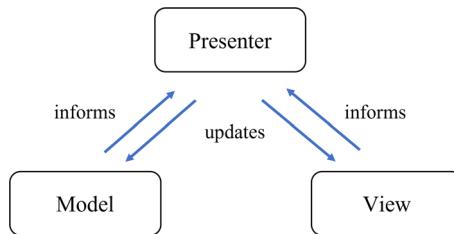


Fig. 5 MVP design

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. 5).

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 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 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 2, the algorithm proposed in 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 results for 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

² <https://github.com/CorinnaKrebs/Instances>.

³ see <https://alim.algorithandexchange.com/orlib/topic/3L-FCVRP/?jsessionid=BE16D2007BD713BBFCCE3A5926C6EFC0/#Zhang2015>.

Table 1 Overview of instance sets

Authors	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]
Zhang et al. (2017)	3L-VRPTW	27	[15, 100]	[26, 199]
Krebs et al. (2021)	3L-VRPTW	600	20, 60, 100	200, 400

Table 2 BKS for Gendreau et al. (2006) instances

No.	ttd	Reference	No.	ttd	References
1	300.69	Escobar-Falcon et al. (2016)	15	1338.22	Zhang et al. (2015)
2	334.96	Zhang et al. (2015)	16	698.61	Zhang et al. (2015)
3	374.81	Escobar-Falcon et al. (2016)	17	866.40	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	Zhang et al. (2015)
7	767.46	Tao and Wang (2015)	21	1067.70	Zhang et al. (2015)
8	804.75	Tao and Wang (2015)	22	1147.80	Bortfeldt (2012)
9	630.13	Zhang et al. (2015)	23	1103.44	Zhang et al. (2015)
10	820.35	Bortfeldt (2012)	24	1102.14	Zhang et al. (2015)
11	772.85	Tao and Wang (2015)	25	1370.34	Zhang et al. (2015)
12	610.23	Zhang et al. (2015)	26	1557.15	Zhang et al. (2015)
13	2608.68	Tao and Wang (2015)	27	1496.28	Zhang et al. (2017)
14	1368.40	Bortfeldt (2012)	Avg	927.15	

the values described in 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.

In Table 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 Zhang et al. (2015). As detailed results are available, these solutions are checked with the Solution Validator and are published via GitHub⁴, except for instance no 34, where a feasible solution could not be found.

Concerning the 3L-VRPTW instances by Zhang et al. (2017), the BKS are presented in Table 4. All solutions are found by the algorithms described in Krebs et al. (2023). As before, the validated results are published via GitHub⁴.

Table 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 GitHub⁴.

⁴ <https://github.com/CorinnaKrebs/BestKnownResults>.

Table 3 BKS for Tarantilis et al. (2009) instances

No.	<i>ttd</i>	References	No.	<i>ttd</i>	References
28	1417.88	Zhang et al. (2015)	34	2595.22	Bortfeldt (2012)
29	2189.27	Zhang et al. (2015)	35	4163.02	Zhang et al. (2015)
30	1713.82	Zhang et al. (2015)	36	3400	Zhang et al. (2015)
31	2010.58	Zhang et al. (2015)	37	3159.15	Zhang et al. (2015)
32	2971.58	Zhang et al. (2015)	38	5315.97	Zhang et al. (2015)
33	2339.3	Zhang et al. (2015)	39	4031.62	Zhang et al. (2015)
			Avg	2942.28	

Table 4 BKS for Zhang et al. (2017) instances

Name	<i>v_{used}</i>	<i>ttd</i>	<i>time</i>	Name	<i>v_{used}</i>	<i>ttd</i>	<i>time</i>
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	1311.32	765.60
VRPTWP07	5	324.29	22.06	VRPTWP21	16	1189.80	1943.54
VRPTWP08	6	320.75	21.19	VRPTWP22	18	1466.27	305.06
VRPTWP09	9	458.32	19.40	VRPTWP23	17	1325.73	690.63
VRPTWP10	7	487.60	58.50	VRPTWP24	16	1289.15	708.50
VRPTWP11	7	493.58	114.46	VRPTWP25	20	1432.66	3600.00
VRPTWP12	9	575.04	14.76	VRPTWP26	24	1642.74	1455.63
VRPTWP13	6	452.05	467.32	VRPTWP27	22	1597.13	2111.58
VRPTWP14	8	550.16	89.08	Total	295	20,894.11	12,728.66

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

Table 5 BKS for Krebs et al. (2021) instances

n	m	Types	Sum v_{used}	Sum ttd	Avg Time
20	200	3	73	8163.90	767.16
		10	71	8351.98	1804.59
		100	73	8444.32	1817.26
	400	3	88	8834.64	2340.33
		10	95	9215.81	2673.61
		100	106	9694.34	3172.97
	60	3	430	39,940.64	1563.98
		10	448	40,562.89	2027.61
		100	464	41,765.02	2163.44
	400	3	693	53,718.93	2287.25
		10	719	55,179.40	2563.58
		100	782	59,381.69	2964.56
100	200	3	465	47,770.81	2024.01
		10	510	50,906.28	2330.88
		100	548	54,124.17	2146.65
	400	3	797	66,799.70	2702.32
		10	865	71,842.97	2937.54
		100	878	72,187.87	2967.32
	Total		8105	706,885.36	2331.14

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.

Declarations

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

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

Appendix

```

Name          31_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          0
1           37        52      1        0          0          0          0          7          1050
...
15          36        16      3        0          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

```

Fig. 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: mlfifo)
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)

```

Fig. 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       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

```

Fig. 8 Exemplary solution file

References

- Baker, B., Coffman, E., & Rivest, R. (1980). Orthogonal packings in two dimensions. *SIAM Journal on Computing*, 9(4), 846–855. <https://doi.org/10.1137/0209064>
- Bischoff, E. E. (2003). Dealing with load bearing strength considerations in container loading problems
- Bischoff, E. E., & Ratcliff, M. S. W. (1995). Issues in the development of approaches to container loading. *Omega*, 23(4), 377–390. [https://doi.org/10.1016/0305-0483\(95\)00015-G](https://doi.org/10.1016/0305-0483(95)00015-G)

- Bortfeldt, A. (2012). A hybrid algorithm for the capacitated vehicle routing problem with three-dimensional loading constraints. *Computers and Operations Research*, 39(9), 2248–2257. <https://doi.org/10.1016/j.cor.2011.11.008>
- Bortfeldt, A., & Yi, J. (2020). The split delivery vehicle routing problem with three-dimensional loading constraints. *European Journal of Operational Research*, 282(2), 545–558. <https://doi.org/10.1016/j.ejor.2019.09.024>
- Ceschia, S., Schaefer, A., & Stützle, T. (2013). Local search techniques for a routing-packing problem. *Computers and Industrial Engineering*, 66(4), 1138–1149. <https://doi.org/10.1016/j.cie.2013.07.025>
- Escobar-Falcon, L. M., Álvarez-Martinez, D., Granada-Echeverri, M., et al. (2016). A matheuristic algorithm for the three-dimensional loading capacitated vehicle routing problem (3L-CVRP). Revista Facultad de Ingeniería Universidad de Antioquia (pp. 09–20). http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S0120-62302016000100002&nrm=iso
- Fuellerer, G., Doerner, K. F., Hartl, R. F., et al. (2010). Metaheuristics for vehicle routing problems with three-dimensional loading constraints. *European Journal of Operational Research*, 201(3), 751–759. <https://doi.org/10.1016/j.ejor.2009.03.046>
- Gendreau, M., Iori, M., Laporte, G., et al. (2006). A Tabu search algorithm for a routing and container loading problem. *Transportation Science*, 40(3), 342–350. <https://doi.org/10.1287/trsc.1050.0145>
- Iori, M., Salazar González, J. J., & Vigo, D. (2007). An exact approach for the vehicle routing problem with two-dimensional loading constraints. *Transportation Science*, 41, 253–264. <https://doi.org/10.1287/trsc.1060.0165>
- Junqueira, L., Morabito, R., Yamashita, D. S., et al. (2013). *Optimization models for the three-dimensional container loading problem with practical constraints* (pp. 271–293). New York: Springer. <https://doi.org/10.1007/978-1-4614-4469-5>
- Koch, H., Bortfeldt, A., & Wäscher, G. (2018). A hybrid algorithm for the vehicle routing problem with backhauls, time windows and three-dimensional loading constraints. *OR Spectrum*, 40(4), 1029–1075. <https://doi.org/10.1007/s00291-018-0506-6>
- Krebs, C., & Ehmke, J. F. (2021a). Axle weights in combined vehicle routing and container loading problems. *EURO Journal on Transportation and Logistics*, 10(100), 043. <https://doi.org/10.1016/j.ejtl.2021.100043>
- Krebs, C., & Ehmke, J. F. (2021b). Vertical stability constraints in combined vehicle routing and 3d container loading problems. In M. Mes, E. Lalla-Ruiz, & S. Voß (Eds.), *Computational logistics* (pp. 442–455). Springer International Publishing.
- Krebs, C., Ehmke, J. F., & Koch, H. (2021). Advanced loading constraints for 3d vehicle routing problems. *OR Spectrum*. <https://doi.org/10.1007/s00291-021-00645-w>
- Krebs, C., Ehmke, J. F., & Koch, H. (2023). Effective loading in combined vehicle routing and container loading problems. *Computers and Operations Research*, 149(105), 988. <https://doi.org/10.1016/j.cor.2022.105988>
- Lodi, A., Martello, S., & Vigo, D. (1999). Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems. *INFORMS Journal on Computing*, 11(4), 345–357. <https://doi.org/10.1287/ijoc.11.4.345>
- Mack, D., Bortfeldt, A., & Gehring, H. (2004). A parallel hybrid local search algorithm for the container loading problem. *International Transactions in Operational Research*, 11(5), 511–533. <https://doi.org/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* (pp. 187–201). Wiesbaden: Gabler. <https://doi.org/10.1007/978-3-8349-9777-7>
- Moura, A., & Oliveira, J. F. (2009). An integrated approach to the vehicle routing and container loading problems. *OR Spectrum*, 31(4), 775–800. <https://doi.org/10.1007/s00291-008-0129-4>
- Pace, S., Turky, A., Moser, I., et al. (2015). Distributing fibre boards: A practical application of the heterogeneous fleet vehicle routing problem with time windows and three-dimensional loading constraints. *Procedia Computer Science*, 51, 2257–2266.
- Potel, M. (1996). Mvp: Model-view-presenter the taligent programming model for c++ and java
- Ramos, A., Silva, E., & Oliveira, J. (2017). A new load balance methodology for container loading problem in road transportation. *European Journal of Operational Research*. <https://doi.org/10.1016/j.ejor.2017.10.050>
- Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2), 254–265. <https://doi.org/10.1287/opre.35.2.254>
- Tao, Y., & Wang, F. (2015). An effective tabu search approach with improved loading algorithms for the 3l-cvpr. *Computers and Operations Research*, 55, 127–140. <https://doi.org/10.1016/j.cor.2013.10.017>
- Tarantilis, C. D., Zachariadis, E. E., & Kiranoudis, C. T. (2009). A hybrid metaheuristic algorithm for the integrated vehicle routing and three-dimensional container-loading problem. *IEEE Transactions on Intelligent Transportation Systems*, 10(2), 255–271. <https://doi.org/10.1109/TITS.2009.2020187>

- Zhang, D., Cai, S., Ye, F., et al. (2017). A hybrid algorithm for a vehicle routing problem with realistic constraints. *Information Sciences*, 394–395, 167–182. <https://doi.org/10.1016/j.ins.2017.02.028>
- Zhang, Z., Wei, L., & Lim, A. (2015). An evolutionary local search for the capacitated vehicle routing problem minimizing fuel consumption under three-dimensional loading constraints. *Transportation Research Part B: Methodological*, 82, 20–35. <https://doi.org/10.1016/j.trb.2015.10.001>

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