An optimization approach for a complex real-life container loading problem

Mikele Gajda[a], Alessio Trivella[b,1], Renata Mansini[c], David Pisinger[d]

[a]Department of Operations, HEC Lausanne, 1015 Lausanne, Switzerland
[b]Institute for Transport Planning and Systems, ETH Zürich, 8092 Zürich, Switzerland
[c]Department of Information Engineering, University of Brescia, 25123 Brescia, Italy
[d]DTU Management, Technical University of Denmark, 2800 Kongens Lyngby, Denmark

We consider a real-world packing problem faced by a logistics company that loads and ships hundreds of trucks every day. For each shipment, the cargo has to be selected from a set of heterogeneous boxes. The goal of the resulting container loading problem (CLP) is to maximize the value of the cargo while satisfying a number of practical constraints to ensure safety and facilitate cargo handling, including customer priorities, load balancing, cargo stability, stacking constraints, positioning constraints, and limiting the number of unnecessary cargo move operations during multi-shipment deliveries. Although some of these constraints have been considered in the literature, this is the first time a problem tackles all of them jointly on real instances. Moreover, differently from the literature, we treat the unnecessary move operations as soft constraints and analyze their trade-off with the value maximization. As a result, the problem is inherently multi-objective and extremely challenging. We tackle it by proposing a randomized constructive heuristic that iteratively combines items in a preprocessing procedure, sorts them based on multiple criteria, uses randomization to partially perturb the sorting, and finally constructs the packing while complying with all the side constraints. We also propose dual bounds based on CLP relaxations. On large-scale industry instances, our algorithm runs in a few seconds and outperforms (in terms of value and constraints handling) both the solutions constructed manually by the company and those provided by a commercial software. The algorithm is currently used by the company generating significant economic and CO_2 savings.

Keywords: Container loading problem; real-life constraints; randomized constructive heuristic; industry collaboration.

1. Introduction

Logistics and transportation systems play a critical role in our increasingly connected economy where speed and efficiency in delivering goods are central themes. According to the Pitney Bowes Parcel Shipping Index, in 2018 there were on average 23 parcels shipped per person in 13 major markets with a total population of 3.7 billion[2], corresponding to 2760 parcels shipped every second (PitneyBowes 2020). The number of parcels shipped globally in 2019 exceeded 100 billion and is expected to hit 200 billion in 2025 (Statista 2020). This forecast does not even account for the

[1]Corresponding author. E-mail: alessio.trivella@ivt.baug.ethz.ch
[2]U.S., Canada, Brazil, Germany, UK, France, Italy, Norway, Sweden, China, Japan, Australia, and India.

Covid-19 pandemic, which is boosting parcel shipment due to the increasing use of e-commerce. For example, SDC (2020) reports a 13% increase in the number of parcels shipped in April 2020.

This trend is an opportunity for logistics and transportation companies to expand their businesses. However, handling an increasing amount of goods and moving them faster is also a challenge since performing these operations efficiently and actively reducing costs is necessary for the companies to remain competitive. The Annual State of Logistics Report of the 31st Council of Supply Chain Management Professionals (CSCMP) tracked an increase in logistics costs (including management, vehicle maintenance, and storage costs) by 0.6% in the U.S. in 2019, reaching \$1.63 trillion, which represents 7.6% of the U.S. total gross domestic product for that year (Kearney 2020). This increment follows the same trend observed in previous years and reflects a rising demand and hence investments in the field. In addition to direct costs, the environmental impact from logistics and trucking is a concern. According to a study from the World Economic Forum, the number of delivery vehicles will increase by 36% in 100 main cities globally by 2030, leading to a 30% CO_2 emission increase from these vehicles and 21% more urban congestion (WEF 2020). In light of these factors and trends, optimizing logistics processes is evermore important to improve service quality and handle more goods while at the same time reducing costs and emissions.

Motivated by ITLM Group, a multinational transportation and logistics company (ITLM 2021), in this paper we consider the important logistics process of loading trucks with goods. Specifically, the company needs to load hundreds of trucks every day for delivery to business customers across Europe. For each shipment, the cargo must be built by selecting items (boxes) from an heterogeneous set available at a warehouse (potentially belonging to multiple customers) and then loaded on a truck. The selected cargo has to maximize the total value of the items, corresponding in our case to the chargeable weight (henceforth *taxability*), which is a function of weight and volume. At the same time, the loaded truck must comply with a number of constraints set either internally by the company or by freight regulations in order to ensure safe transportation and facilitate cargo handling. These constraints include loading priorities, unloading order, load balancing, cargo stability, load-bearing, weight limits, and various other positioning constraints. Clearly, our collaborating company highly benefits from effectively selecting and loading the boxes inside trucks by increasing its revenues from shipments. In turn, this generates savings on the leasing of the containers and/or trucks by requiring a smaller number of them, and decreases fuel consumption, thus reducing the company's carbon footprint. In addition to the efficient utilization of the truck's internal space, correctly distributing the load and balancing cargo in agreement with the freight regulations is necessary to improve safety, avoid damaging cargo and exclude fines due to irregular loading.

Before our involvement, the company made all loading decisions manually. Despite the experience of the warehouse personnel, the loading procedure could at times be affected by serious evaluation errors resulting in badly optimized trucks, e.g., in terms of taxability or weight distribution. In fact, having many constraints to satisfy and a large number of different size and weight boxes from which to select the cargo makes this procedure extremely tricky to be carried out manually. For example, the initial loading pattern that an employee has in mind often changes during the loading phase once it is evident the initial evaluation is unsatisfactory, resulting in some extra time to fix it. Moreover, it is especially difficult to guarantee optimal load balancing and a safe distribution of weight along the truck's axles when the procedure is done manually without any analytical support. The company also tested several commercial software like the well-known EasyCargo (2020), but their lack of flexibility to set up proprietary constraints pushed the company to create its own tool based on an academic collaboration.

The considered problem is commonly known as a container loading problem (CLP), a three-dimensional packing problem in which a set of items (usually rectangular-shaped objects/boxes) must be loaded into a larger container according to some predefined rules. Since not all available items can be loaded into the container, a subset of them has to be selected while maximizing their value (represented by the volume, or the profit, or the taxability as in our case). In addition to being $\mathcal{NP}$-hard, a major challenge in dealing with a CLP stems from the number and nature of constraints that a loaded container must satisfy and that we review in Section 2. Given its wide applicability, it is not surprising that the CLP constitutes an active area of research and that many problem variants as well as solution methods have been proposed in the literature. However, we notice that: (i) the constraints considered in many method-oriented papers do not really reflect the practical needs of companies, (ii) most works focus on the single objective to maximize volume utilization, and (iii) no method in the literature handles the variety and combination of constraints we consider. In this paper, we overcome all indicated drawbacks. In particular, all practical constraints in our problem have been specified with ITLM Group after a site visit lasting several days, and perfectly define its needs. Moreover, while the maximization of taxability is considered the primary objective, we explore the underlying multi-objective nature of the problem by means of soft constraints. For example, the packing solution should limit the number of unnecessary movement of items during unloading, which would result in extra time and work for the driver and hence a cost for the company. We refer to those boxes that need to be relocated as *unloading obstacles* and the corresponding unloading constraints introduce a threshold on their number. In the literature on vehicle routing, unloading constraints are also known as *sequential constraints* and treated as hard constraints (see Pollaris et al. 2015 and references therein). Instead, we allow (but penalize) obstacles, which is preferable in practice since it leads to a better utilization of the vehicle capacity and limiting their number can be treated as an additional goal. In fact, we analyze in this paper the trade-off between the maximization of the cargo taxability and the minimization of unloading obstacles, providing approximated Pareto frontiers that help practitioners in selecting the preferred solution.

Given the complexity and number of constraints involved, we tackled our CLP by developing a randomized constructive heuristic algorithm aiming to produce good packing solutions quickly. Our algorithm consists of multiple phases: preprocessing, items sorting, list randomization, and packing construction, which are performed iteratively in a multi-run approach. Specifically, given a list of strongly heterogeneous items, each one provided with a set of features (dimensions, weight, priority level, taxability, customer number, etc.), the algorithm first tries to combine boxes in larger ones (preprocessing phase). Then, it arranges the resulting items in a sorted list based on several features/criteria. After sorting, the list is perturbed by exchanging the position of some items according to a set of probabilistic rules. Then, items are packed one by one using a constructive method that exploits specific positions inside the container so that all constraints can be satisfied, while minimizing the number of unloading obstacles is treated as a second objective. By iterating between preprocessing, sorting, randomization, and packing, we produce different loading solutions and ultimately choose a set of non-dominated solutions in terms of taxability and number of unloading obstacles (to compute such a number we simulate the cargo delivery). We also visually output such solutions in 3D so that the warehouse personnel can select the preferred loading pattern to implement: sometimes slightly worse solutions may actually be easier to load, reducing time and effort by the workers.

We tested our algorithm on real instances from the company. Compared to their original loading decisions, our solutions comply with all hard constraints and better satisfy soft constraints while increasing taxability by 7.7% on average across all instances. If we exclude the instances in which the company's solution produces a cargo that violates the weight limits, the average taxability improvement of our method reaches 22.4%. The realized improvement shows a positive impact in transportation efficiency and loading operations. Besides, by using a system based on our algorithm, the company estimates annual savings of around one million Euros and a reduction in CO_2 emissions by one thousands tons. Our solutions also exhibit higher taxability than those generated by the tool EasyCargo, despite our solutions are more constrained (i.e., not all constraints we deal with can be enforced in EasyCargo). We also benchmark our taxability values to an upper (dual) bound based on different CLP relaxations, showing that in several instances our solution is optimal or close to optimality. We finally tested our method on 1500 benchmark instances of a well-studied problem variant that maximizes cargo volume and excludes almost all practical constraints. Although this problem is much simplified and far from the one our method has been developed for, we found that our method performs well and achieves an average container volume utilization of 83–90%.

The rest of this paper is organized as follows. In Section 2, we review the relevant literature. In Section 3, we define in detail the problem and all its constraints. We describe our heuristic algorithm in Section 4, and present a numerical study in Section 5. We conclude in Section 6.

2. Literature review

Like the bin-packing, knapsack, and cutting stock problem, the CLP belongs to the class of *Cutting & Packing* problems. These problems share a similar high-level structure where a set of items must be packed into one or more objects of larger size (containers), but differ in terms of the number of containers used, dimensions, objective function, and constraints considered (Wäscher et al. 2007).

The CLP is a three-dimensional packing problem that involves filling a single container. In most applications, both the container and the items to pack are rectangular-shaped/cuboids and the packing is required to be orthogonal, i.e., the surfaces of all items need to be parallel either to the floor or to a wall of the container. Moreover, items inside the container cannot obviously overlap in space. The basic CLP that only accounts for these standard features is essentially a 3D knapsack problem (3DKP). Motivated by practical applications, additional specific constraints have been incorporated into the CLP resulting in a variety of heuristic and exact solution methods developed to solve it. We divide our literature review into three parts. Since the 3DKP with rotation is a relaxation of the considered problem and is useful to obtain dual bounds, we present its mathematical formulation in Section 2.1. We then focus in Section 2.2 on the constraints studied in previous works and, in Section 2.3, review the main solution approaches developed for the CLP.

2.1 Basic CLP formulation

Given a container and a set of items, the basic CLP (or 3DKP) aims at selecting and orthogonally loading in the container the subset of items that maximize the total value or profit. More formally, we are given a container with the three dimensions length, width, and height equal to $(L, W, H) \in \mathbb{R}^3_+$ and a set $\mathcal{B}$ of boxes, where box $i \in \mathcal{B}$ has size $(l_i, w_i, h_i) \in \mathbb{R}^3_+$ in its original orientation, weight $q_i > 0$, and value $\pi_i > 0$. The orientation and dimensions of an empty container are introduced in Figure 1, where the eight corners are identified by a sequence of three letters: (i) "F" front of "R" rear, (ii) "L" left or "R" right, and (iii) "B" bottom or "T" top. The rear side corresponds to the

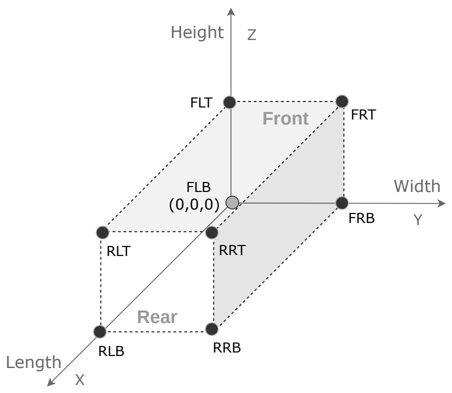


Figure 1: Container dimensions and orientations.

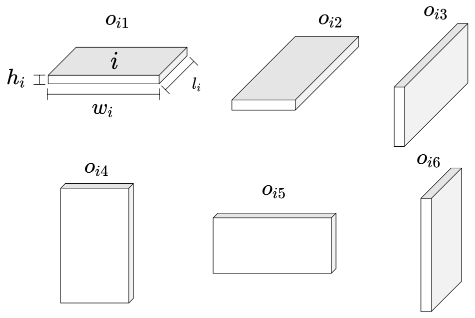


Figure 2: Six orthogonal item orientations.

back doors of the truck from which unloading takes place. A box can be placed into the container in any orientation which maintains its faces parallel to the container faces (this means six different spatial rotations for each box as shown in Figure 2). The 3DKP with rotation can be formulated as the following mixed-integer linear program (MIP) (Baldi et al. 2012):

$$
\begin{aligned}
\max \quad & \sum_{i \in \mathcal{B}} \pi_i \, t_i && && \text{(1a)}\\
\text{s.t.}: \quad & f_{ij} + f_{ji} + b_{ij} + b_{ji} + u_{ij} + u_{ji} + (1 - t_i) + (1 - t_j) \geqslant 1 && \forall i, j \in \mathcal{B},\ i < j, && \text{(1b)}\\
& x_i + w_i(o_{i2} + o_{i4}) + l_i(o_{i1} + o_{i6}) + h_i(o_{i3} + o_{i5}) - x_j \leqslant L(1 - b_{ij}) && \forall i, j \in \mathcal{B}, && \text{(1c)}\\
& y_i + w_i(o_{i1} + o_{i3}) + l_i(o_{i2} + o_{i5}) + h_i(o_{i4} + o_{i6}) - y_j \leqslant W(1 - f_{ij}) && \forall i, j \in \mathcal{B}, && \text{(1d)}\\
& z_i + w_i(o_{i5} + o_{i6}) + l_i(o_{i3} + o_{i4}) + h_i(o_{i1} + o_{i2}) - z_j \leqslant H(1 - u_{ij}) && \forall i, j \in \mathcal{B}, && \text{(1e)}\\
& x_i + w_i(o_{i2} + o_{i4}) + l_i(o_{i1} + o_{i6}) + h_i(o_{i3} + o_{i5}) \leqslant L && \forall i \in \mathcal{B}, && \text{(1f)}\\
& y_i + w_i(o_{i1} + o_{i3}) + l_i(o_{i2} + o_{i5}) + h_i(o_{i4} + o_{i6}) \leqslant W && \forall i \in \mathcal{B}, && \text{(1g)}\\
& z_i + w_i(o_{i5} + o_{i6}) + l_i(o_{i3} + o_{i4}) + h_i(o_{i1} + o_{i2}) \leqslant H && \forall i \in \mathcal{B}, && \text{(1h)}\\
& o_{i1} + o_{i2} + o_{i3} + o_{i4} + o_{i5} + o_{i6} = 1 && \forall i \in \mathcal{B}, && \text{(1i)}\\
\text{var.}: \quad & f_{ij},\, b_{ij},\, u_{ij},\, t_i,\, o_{i1},\, o_{i2},\, o_{i3},\, o_{i4},\, o_{i5},\, o_{i6} \in \{0, 1\} && \forall i, j \in \mathcal{B}, && \text{(1j)}\\
& x_i,\, y_i,\, z_i \geqslant 0 && \forall i \in \mathcal{B}. && \text{(1k)}
\end{aligned}
$$

The decision variables (x_i, y_i, z_i) represent the coordinates of the front-left-bottom corner of box i with respect to the origin of the axes represented by the `FLB` corner of the container. Binary variables o_{i1}, o_{i2}, o_{i3}, o_{i4}, o_{i5}, o_{i6} determine the six possible orientations (see Figure 2), whereas variables (f_{ij}, b_{ij}, u_{ij}) indicate the relative position (left, behind, and under) of box i with respect to box j and are used to model overlapping. Finally, variables t_i establish whether boxes are selected or not.

The objective (1a) maximizes the profit of the loaded boxes. Constraints (1b) enforce no overlap between each pair of loaded items, (1d)–(1c) define the overlapping conditions, (1g)–(1f) ensure that loaded items are fully inside the container, and (1i) enforces boxes to only have one orientation. Finally, (1j)–(1k) define the domain of the variables. Notice that the 3DKP without rotation can be defined from (1) by fixing the orientation of each box i, i.e., by setting $o_{i1} = 1$.

Model (1) contains $\mathcal{O}(|\mathcal{B}|^2)$ binary variables and $\mathcal{O}(|\mathcal{B}|^2)$ big-M constraints, and is extremely difficult to solve to optimality for large-sized instances, as also confirmed by the comparative study by Silva et al. (2019). Nevertheless, this model is practically useful as it allows to derive dual bounds on the maximum cargo value also to container loading problems with more practical constraints.

2.2 Practical CLP constraints

The basic CLP (or 3DKP) only accounts for standard "physical" constraints (non-overlapping, fitting within the space). In real-life applications such as the loading of trucks or aircrafts, more practical limitations need to be considered, making the CLP harder to formulate and solve. This has already been noticed by Bischoff and Ratcliff (1995), who underlined the fact that new approaches were necessary to handle CLPs with more practically useful constraints. In Table 1, we classify the most important of such constraints into five categories following Bortfeldt and Wäscher (2013).

Table 1: Classification of CLP constraints.

Container-related	Item-related	Cargo-related	Positioning-related	Load-related
▷ Weight limits ▷ Weight distribution	▷ Loading priorities ▷ Orientations ▷ Stacking	▷ Complete shipment ▷ Allocation	▷ Dangerous items ▷ Relative positioning ▷ Multi-drop situations	▷ Stability ▷ Complexity

Container-related constraints in the first group are widely used and include weight limits imposed on the container/vehicle used for transportation and the distribution of weight. The weight limit is a structural property that depends on the container's characteristics and is commonly enforced as a hard constraint (Wang et al. 2010, Ramos et al. 2018). The weight distribution – also referred to in the literature as *load balancing* – has been treated in different ways, for example, by introducing a "stable region" where the center of mass of the cargo must lie (i.e., a hard constraint; see Baldi et al. 2012, Ramos et al. 2018) or by pushing the center of mass as close as possible to an ideal location (i.e., a soft constraint; see Trivella and Pisinger 2016). Load balancing can be considered either along the longitudinal axis of the container only (Lim et al. 2013), or both longitudinal and traversal axes (Bortfeldt and Gehring 2001, Moon and Nguyen 2014), or jointly longitudinal, traversal, and vertical axes (Egeblad and Pisinger 2009, Baldi et al. 2012, Trivella and Pisinger 2016).

The second group (*item-related* constraints) describes properties and limitations of individual items such as loading priorities, orientations, and stackability. When not all items can fit within the available space, loading priorities influence the selection and can be a mix of hard and soft constraints, that is, some items may be compulsory to load while others are treated based on their level of urgency. Priorities may reflect the time-sensitivity of the goods to transport or the customer type (Junqueira et al. 2012, Jamrus and Chien 2016). Orientation constraints limit the set of feasible rotations of an item inside the container, among the six possible orthogonal orientations shown in Figure 2 (Amossen and Pisinger 2010, Fanslau and Bortfeldt 2010). Frequently, the base of the box and hence its height are fixed ("this way up") but it is also possible that more rotations are allowed or that different items to pack have different freedom of being rotated. Stacking constraints, often called *load-bearing* constraints, restrict how boxes can or cannot be placed on top of each other based on the amount of weight that each of them can bear on its top without being damaged. In practice, it is hard to define a maximum bearable weight for each item and a simplified classification into fragile and non-fragile items is often used (Gendreau et al. 2006, Tarantilis et al. 2009). If items can be rotated, the weight applicable on top may vary across orientations, adding complexity to the problem.

The third group comprises *cargo-related* constraints like complete shipment and cargo allocation, which have rarely been considered in the literature. Complete shipment constraints refer to sets of items that cannot be split, i.e., either the set is fully loaded in the same container or it is not (Eley 2003). Allocation constraints define instead classes of items that cannot be placed in the same container as they would negatively affect each other's quality. For instance, food and perfume articles or petroleum are not allowed to be loaded in the same container (Bortfeldt and Wäscher 2013, Colombi et al. 2017).

Positioning-related constraints in the fourth group refer to objects that must be placed in specific positions inside the container. For example, packages containing flammable goods and in general dangerous items (known as "ADR") must be placed by law close to the side of unloading. Also in this group are relative positioning (or grouping) constraints, requiring some packages to be loaded close to each other (Egeblad et al. 2010), and multi-drop situations, in which the shipment contains items for multiple customers at different locations. In this case, it is preferable that items for the same customer are packed next to each other and that subsets of items for different customers respect the sequence of delivery to avoid extra unloading and reloading operations due to unloading obstacles. Silva et al. (2018) account for multi-drop situations by rebalancing the cargo after portions of it have been unloaded, but do not attempt to avoid unloading obstacles beforehand when loading the truck. As discussed in Section 1, the literature on unloading constraints has always treated them as hard constraints to our knowledge (Pollaris et al. 2015), which reduces flexibility while packing.

The last group of constraints (*load-related*) describes preferences or rules on the final disposition of objects inside the container. The stability of the loaded vehicle is one of the most important aspects in container loading (Bischoff 1991, Moura and Oliveira 2005). A lack of cargo stability may in fact lead to items damage during transportation, with a consequent loss for the company. In the literature, authors either consider *static stability* by requiring all boxes to satisfy the static equilibrium theory (Oliveira et al. 2020), or more commonly, model *vertical stability* by using the support factor approach (Junqueira et al. 2012, Kurpel et al. 2020). Despite its importance, load stability is not often imposed explicitly in the literature as many authors believe that compact solutions automatically lead to a stable load. In practice, stability may also be ensured by filling gaps with filler material (Pisinger 2002, Parreño et al. 2008). In the same category, complexity constraints refer to the difficulty of a packing disposition to be understood and realized concretely by the loading personnel without excessive effort. An example of complexity constraints are the so-called *guillotine-cutting* constraints (Amossen and Pisinger 2010).

Some recent works incorporate practical constraints in the CLP but only consider a small subset of the constraints described above. For example, Alonso et al. (2019) consider maximum weight on axles, the position of the center of mass, and stability. Kurpel et al. (2020) account for box orientations, stability, and separation of boxes. Nascimento et al. (2021) consider instead many practical constrains, but only model and test one constraint at a time, i.e., the constraints are not taken jointly. Another recent work by da Silva et al. (2020) incorporates several practical constraints but deals with weakly heterogeneous items, and the solution method heavily relies on this assumption, while our instances can be strongly heterogeneous. In conclusion, not many papers on CLP account for a significant set of practical constraints jointly and on general and large-scale instances. Even fewer papers tested the proposed CLP variants and methodologies on industry data.

2.3 Solution methods

Although the existing literature has tackled the CLP by using both exact methods and heuristic or meta-heuristic algorithms, the latter are significantly more widespread for several reasons. First, the CLP is known to be $\mathcal{NP}$-hard (Pisinger 2002). Second, although the basic CLP can be formulated as an MIP (see Section 2.1), it is difficult or even impossible to formulate some of the practical constraints in the same form. Third, even the basic CLP model presents symmetries and big-M constraints that make linear programming relaxations weak; additional constraints may exacerbate these complicating features. Fourth, logistics companies operate on highly time-sensitive processes and require a quick generation of packing patterns, which is unattainable by exact methods.

Silva et al. (2019) present a comparative study of exact approaches for cutting and packing problems. Using the basic CLP as a test case, the authors examine different mathematical models developed for this problem (Martello et al. 2000, Fasano 2004, Hifi et al. 2010, Junqueira et al. 2012, Tsai et al. 2015, Paquay et al. 2016); after 15,000 hours of testing, they state: *"results show how poorly these methods perform, not being able to solve to optimality challenging instances with less than 20 boxes over 6h of runtime"*. Even though commercial solvers are constantly improving and recent works have extended the CLP mathematical program to handle load balancing (Trivella and Pisinger 2016, Ocloo et al. 2020), stability (Alonso et al. 2017, Kurpel et al. 2020), and a few other constraints (Alonso et al. 2019, Nascimento et al. 2021), exact methods remain far from meeting industry standards where large instances with multiple constraints must be solved quickly. For example, most papers relying on MIPs consider rather small instances and test them with time limit of 3600 seconds, often reached (Alonso et al. 2019, Kurpel et al. 2020, Nascimento et al. 2021).

Beyond exact formulations, a variety of heuristic approaches have been proposed in the literature to solve large instances in a limited amount of time and/or accommodate challenging constraints, at the cost of sacrificing optimality. A good review of heuristic methods can be found in Zhao et al. (2016), that compares different packing problems and classifies heuristics into *construction* (or *placement*) methods and *improvement* methods. The former class of methods relies on iteratively placing boxes in specific positions inside the container, while the latter attempts to improve an initial packing solution by performing neighboring moves.

A classic construction method is the wall-building approach, presented for the first time by George and Robinson (1980), where the available space is filled up by *walls* created from boxes with a maximum-defined thickness. Similarly, layer-building strategies produce *layers* of boxes of comparable height and place these layers one on top of the other, from bottom to top of the container (Pisinger 2002, Bischoff 2006, Iori et al. 2020). Layer-building usually ensures more stability to the load compared to wall-building, but works well when boxes have similar dimensions and are stackable one over the other. Related methods pre-assemble items into *stacks* or *blocks* of similar objects and then load these resulting bigger structures (Gehring and Bortfeldt 1997, Bortfeldt et al. 2003, Araya et al. 2017, da Silva et al. 2020). Guillotine cuts are another way of dividing the space inside a container (Fanslau and Bortfeldt 2010). Differently, Lim et al. (2003) propose a multi-faced buildup algorithm where partial solutions are constructed starting from each wall of the container and then combined into a single packing. Some papers developed constructive heuristics that pack individual items sequentially by using candidate points in the container, such as the so-called *corner points* (Martello et al. 2000) or *extreme points* (Crainic et al. 2008), which are related to the placement strategy we use in this paper. Finally, tree-search-based approaches evaluate each box insertion by constructing a search tree of limited size, which allows exploring a portion of solution space from the current partial solution (Fanslau and Bortfeldt 2010, Araya and Riff 2014, Araya et al. 2017). Tree-search methods are shown to be effective on the standard CLP (1); for instance, Araya et al. (2017) report the best-known results for widely used CLP benchmark instances (Bischoff and Ratcliff 1995, Davies and Bischoff 1999). However, such methods are harder to apply when handling the variety of practical constraints considered in this paper. The algorithm we devise adds to the constructive heuristics currently available in the literature for the CLP by accounting for several practical constraints jointly and by doing so extremely quickly. We are also not aware of existing heuristics with the same design (block-construction, sorting, randomization, construction).

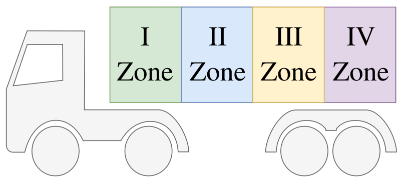
Improvement heuristics are also popular in CLPs and packing problems in general. Starting from a complete packing solution (e.g., obtained by a constructive heuristic), these methods iteratively improve the objective function by exchanging boxes inside the container with boxes outside, or by moving boxes in the container to free space for additional items. Improvement heuristics from the literature include genetic algorithms (Bortfeldt and Gehring 2001, Ramos et al. 2018), local search (Trivella and Pisinger 2016), tabu search methods (Bortfeldt et al. 2003), and various other approaches to deal with specific problem variants or constraints. An improvement phase is also used in GRASP algorithms (Moura and Oliveira 2005, Martínez et al. 2015, Iori et al. 2020). Since the improved phase in the mentioned papers is more computational expensive than the construction phase, we do not consider it our approach, while relying instead on randomized construction.

3. Problem definition

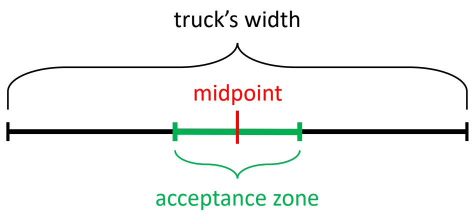
Herein, we describe the tackled problem by relying on the notation introduced in Section 2.1. The aim of our CLP is to maximize the value of the cargo, measured by its taxability (also known as taxable weight or chargeable weight). Formally, the taxability π_i of an item $i \in \mathcal{B}$ is defined as the maximum between its weight q_i (kg) and a volumetric weight computed as the product of its volume $v_i := l_i \cdot w_i \cdot h_i$ (m^3) and a constant α (kg/m^3), that is, $\pi_i = \max\{q_i, \alpha v_i\}$. This means that the taxability of high-density items (e.g., metals) corresponds to their weight whereas the taxability of light and bulky items is determined by their volume. Consequently, maximizing taxability involves managing some trade-offs: given two high-density items of equal volume, we prefer to load the heavier one as it brings more taxability, but this item would also make compliance with weight limits more difficult. Likewise, given two low-density items of equal weight, we would choose the bulkier due to its higher taxability, however, this item also requires more free space for uploading. The following list takes into account all the constraints we consider after establishing them with the company.

(C1) Weight limit. The total weight of the loaded items must not exceed a predetermined constant Q^{MAX}, which depends on the truck's characteristics and is treated as a hard constraint.

(C2) Weight distribution and load balancing. The load inside the truck must be distributed as evenly as possible along the axles. We consider both longitudinal and horizontal load balancing as follows. Longitudinally, the truck is divided into several zones as illustrated in Figure 3(a). The number of zones and their size is defined by the company based on different features of the vehicle, and each zone can support a maximum declared weight. Since zones are not physically separated inside the truck, an item i may lie across two distinct zones (or even more zones in case of long items). To compute the contribution of the item's weight q_i to a zone Z, we assume that each item has a homogeneous density. If item i is placed directly over the floor of the container (i.e., there are no items beneath), then this contribution is $\gamma^{\text{Z}} \cdot q_i$, where γ^{Z} is the proportion of the base area of i ($w_i \cdot l_i$) that lies over zone Z. Otherwise, the weight is distributed over the boxes below proportionally to the size of the contact area. Horizontally,



(a) Longitudinal zone separation.



(b) Horizontal acceptance zone.

Figure 3: Load balancing constraints.

load balancing is instead enforced by requiring that the center of mass of the loaded container does not exceed a maximum displacement from the horizontal midpoint. Figure 3(b) shows an acceptance zone around the horizontal midpoint in which the balance is deemed feasible.

Longitudinal and horizontal load balancing are both hard constraints that mush hold through the entire tour to visit customers, i.e., the cargo must remain balanced after each delivery. Since our definition of longitudinal balancing relies on maximum weight for each truck zone, satisfying this constraint for the whole loaded cargo automatically enforces it throughout the entire tour as items can only be unloaded. However, the center of mass may move outside the acceptance zone during the unloading process; hence, horizontal balancing must be enforced to all partial packing solutions resulting after each customer delivery. Hereafter we indicate the longitudinal and horizontal load balancing constraints by ($\texttt{C2}^{\text{L}}$) and ($\texttt{C2}^{\text{H}}$), respectively.

(C3) Loading priorities. Items have different levels of priority, meaning that some of them must be treated more urgently than others. We denote by p_i the priority of an item $i \in \mathcal{B}$. In our context, we are given 10 priority levels from 1 to 10. Priority 10 is regarded as a hard constraint, meaning that the load of the item is mandatory. For example, in case of partial shipments where the truck reaches the warehouse with some preloaded items, those items can be unloaded and reordered to improve the final packing, but ultimately have to be loaded back onto the same truck. Other priority values from 1 to 9 are treated as soft constraints, where 9 denotes very urgent shipments whereas 1 denotes non-urgent items, as shown in Table 2. As customers are business customers that are served regularly, no complete shipment is imposed by the company, i.e., loading a subset of the items for a given customer is allowed.

Table 2: Priority values and constraints.

p_i	Description	Constraint type
10	Mandatory item	hard
9	Highest level of priority	soft
...	...	soft
1	Lowest level of priority	soft

(C4) Orientation constraints. Each item i is provided with an initial orientation (without loss of generality, o_{i1}) and a subset of the six orthogonal orientations $\{o_{i1}, o_{i2}, o_{i3}, o_{i4}, o_{i5}, o_{i6}\}$ shown in Figure 2, in which the item can be feasibly rotated when packing it.

(C5) Stacking constraints. Each item has a stacking value of 1 (resp. 0), which indicates that the item can (resp. cannot) bear the weight of other items on top of it. In other words, a value of 0 is assigned to fragile items and a value of 1 to non-fragile items. Stackable items are assumed not to have any limitation on the weight they can bear.

(C6) Dangerous cargo. Items that are classified as ADR (Accord européen relatif au transport international des marchandises Dangereuses par Route) must be placed next to an unloading point. This is a hard constraint due to regulation. Although the rear of the vehicle is always considered an unloading point, some types of truck used by the company also allow unloading from one of the lateral sides. We include this feature in our model.

(C7) Stability constraints. We require a packing solution to be vertically stable by using the support factor approach (Junqueira et al. 2012, Oliveira et al. 2020). We model this by enforcing each item to be supported by at least a minimum portion of its base area by underlying items. In other words, the cantilever portion of an item cannot exceed a certain percentage. This is a common approach in the literature, also adopted by our collaborating company. Figure 4 illustrates this concept, showing supported and exceeding area (cantilever portion) of an item.

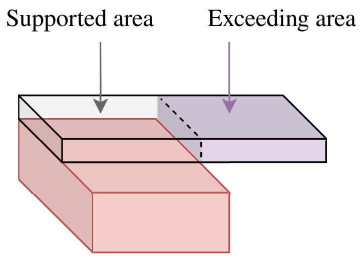
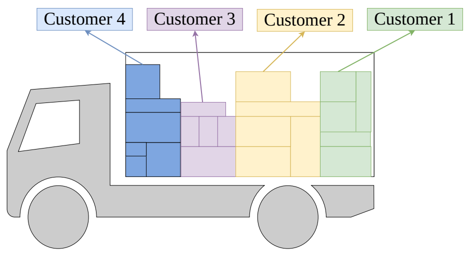


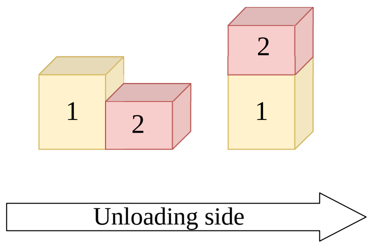
Figure 4: Stability constraints via supported area.

(C8) Multi-drop shipments.

In our instances, trucks often deliver cargo to multiple customers. Each item is then assigned a customer code (or unloading order) $c_i \in \{1, \dots, C\}$, where 1 is the first customer to serve and C is the last. To unload items with minimal effort, items belonging to a given customer should be unloaded without relocating other items that have to be delivered later as shown in Figure 5(a). Since avoiding the extra unloading moves altogether would be too restrictive, we consider this requirement as a soft constraint (basically, a second objective) that we measure by simulating the unloading procedure and counting the total number of extra moves.



(a) Ideal cargo arrangement.



(b) Unloading obstacle.

Figure 5: Example of multi-drop shipment and unloading obstacles.

Formally, an extra move can be defined as a box of a customer j that needs to be moved (unloading obstacle) in order to unload another item for the customer $i < j$ currently being served. As mentioned in Section 1, dealing with extra moves caused by the presence of unloading obstacles is indeed time consuming and hence costly for the company. To illustrate this, Figure 5(b) shows the two most common situations in which items of a customer to be served later (labeled 2) obstruct the unloading of items that needs to be unloaded first (labeled 1).

Given the number and variety of the constraints considered, finding feasible and good solutions to this problem is non-trivial and may require considering an enormous number of packing patterns. Below, we present our solution approach to solve the problem efficiently and to produce a number of different packing solutions for the loading operator to choose from.

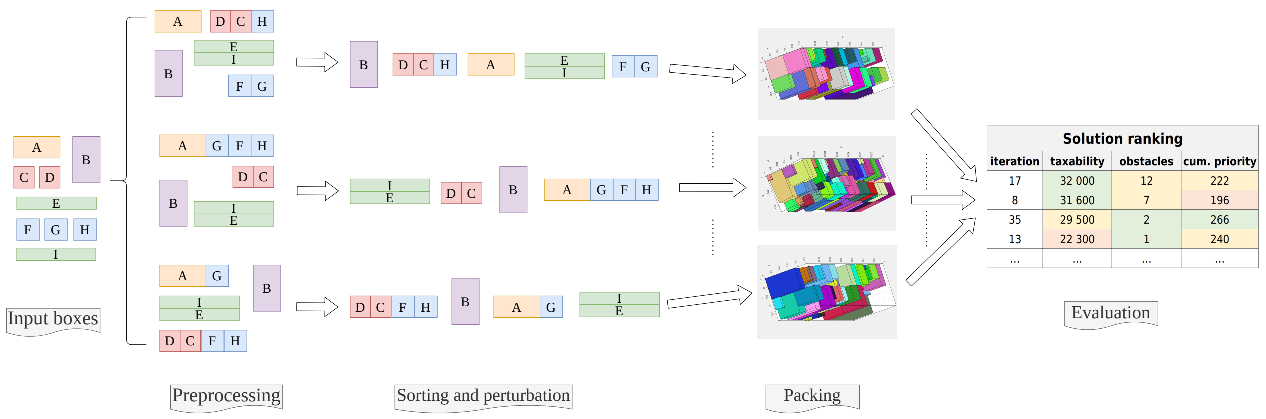
4. A randomized constructive solution approach

Due to the challenging constraints introduced earlier and industry instances to be solved with several hundred items, exact methods are not suitable as discussed in Section 2.3. Moreover, the company requires solutions to be generated in a few seconds, which makes sophisticated heuristics or matheuristics from the literature not applicable to our context. For these reasons, we propose a randomized constructive heuristic method that is able to provide multiple solutions of good quality very quickly. We outline our approach in Section 4.1 and describe each phase in detail from Section 4.2 to 4.5. We finally present upper bounds on taxability based on relaxations of our CLP in Section 4.6.

4.1 Algorithm overview

Our randomized constructive heuristic (RCH) is outlined in Figure 6. After a first preprocessing phase in which items are combined according to their dimensions to construct larger blocks, the resulting items are sorted in a partially randomized list, and finally packed by using a constructive method. The three phases are repeated iteratively and all generated solutions are evaluated. The constructive phase (labeled "Packing" in the figure) takes as input an ordered list of items and inserts them sequentially in the container, starting from the first item in the list, by using specific candidate locations inside the container. Since the ordering of items matters while loading, we initially sort them into a list according to features that make the subsequent packing easier or more profitable, e.g., by placing high-value items at the beginning of the list so that they are considered first and more likely to be loaded. Controlled randomization can be seen as a diversification phase where the initial list order is partially perturbed. This generates different lists that lead to several loading patterns which we finally evaluate based on multiple criteria.

The framework is also presented as a pseudo-code in Algorithm 1, where we indicate the sections in the paper where the different phases are detailed. At each iteration, most of the constraints, i.e.,



Solution ranking

iteration	taxability	obstacles	cum. priority
17	32 000	12	222
8	31 600	7	196
35	29 500	2	266
13	22 300	1	240
...	...	...	...

Figure 6: Overview of RCH and its main phases.

(C1), (C2^{L}), (C4), (C5), (C6), and (C7), are enforced explicitly during the packing phase. Instead, we deal with the soft constraints (C3) and (C8) implicitly during the sorting by placing higher priority items earlier in the list and by grouping items for the same customer. Mandatory items (C3), however, are hard constraints and solutions in which not all such items are loaded are marked as infeasible. The same is done to solutions that do not comply with horizontal load balancing throughout the whole delivery (C2^{H}). In contrast, we do not discard solutions that do not comply with constraint (C8) but consider the number of unloading obstacles as an evaluation criterion. The output of the algorithm is a set of packing solutions, where each solution encodes which items to load, the loading order, and the placement coordinates of each box inside the container.

Given the practical nature of our application, the presented RCH framework has several desirable

Algorithm 1: RCH

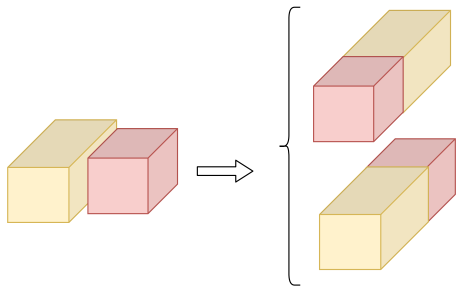
```
Input: CLP instance with set of boxes $\mathcal{B}$; number of iterations $N$; number of solutions $S$
for iteration $n = 1$ to $N$ do
    Preprocess items in $\mathcal{B}$ by combining them into larger blocks [▷ Section 4.2]
    Sort and randomize boxes in $\mathcal{B}$ based on taxability, customer (C8), priority (C3),
      dimension, weight, and orientation (C4), to obtain list $\mathcal{L}_n$ [▷ Section 4.1]
    Construct solution $X_n$ from list $\mathcal{L}_n$ that satisfies weight limit (C1), longitudinal load
      balancing (C2$^{L}$), stacking (C5), dangerous cargo (C6) and stability (C7) [▷ Section 4.4]
    if a mandatory box $i$ is not packed in $X_n$ or horizontal load balancing (C2$^{H}$) is not
      respected throughout the whole delivery, then
        mark $X_n$ as infeasible
Discard infeasible solutions from $\{X_n, n = 1, \ldots, N\}$ and evaluate remaining ones based on
taxability, number of unloading obstacles (C8), and cumulative priority (C3) [▷ Section 4.5]
Output: The $S$ best non-dominated feasible solutions
```

properties. First, RCH is very flexible and can be applied to complex instances such as strongly heterogeneous item sets. In fact, our method does not rely on procedures that exploit the presence of identical items (as done e.g. in da Silva et al. 2020). Avoiding assumptions on item dimensions makes our heuristic more flexible to solve instances ranging from homogeneous to strongly heterogeneous. Second, it outputs not only one solution but multiple solutions corresponding to different loading patterns. As a result, the warehouse staff can choose which pattern to load, potentially sacrificing some taxability but making loading operations easier and faster. Third, the algorithm is extremely quick. Indeed each iteration involves $\mathcal{O}(|\mathcal{B}|^2)$ operations (as we will show later) and takes a split second to run even in large instances. The whole RCH algorithm hence runs in $\mathcal{O}(N|\mathcal{B}|^2)$, but the N iterations could also be executed in parallel as they are completely independent.

4.2 Preprocessing

To handle more efficiently large-scale instances, we introduce a preprocessing phase with the goal to combine items that share certain characteristics into larger blocks. Such a preprocessing is useful to better fill the truck volume, and eases the subsequent phases of the algorithm by reducing the number of items to be loaded. At each iteration of the algorithm, the preprocessing decides among three possible strategies: either it keeps the set of items as it is (i.e., the preprocessing phase is skipped), or it applies one of the two strategies described below for building larger blocks. These three alternatives have the same probability to be chosen at each iteration n of RCH.

1. *Combine items sharing two dimensions*: if two items in $\mathcal{B}$ share two out of three dimensions (length, width, height), they are combined with equal probability into one of the two blocks shown in Figure 7(a). Although the two possible blocks share all dimensions, they differ by their weight composition because the individual items inside the block may have different density. After all items in $\mathcal{B}$ have been considered, the procedure is repeated iteratively for a predefined number of times, so that blocks made of multiple items can be created.


(a) Items sharing two dimensions.

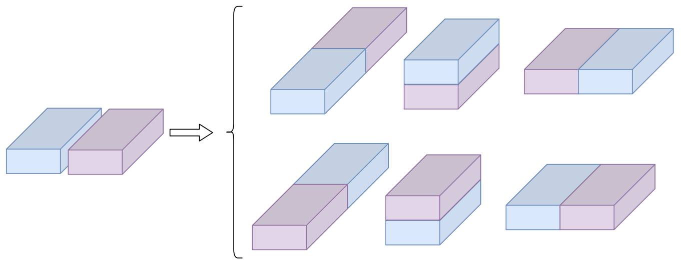
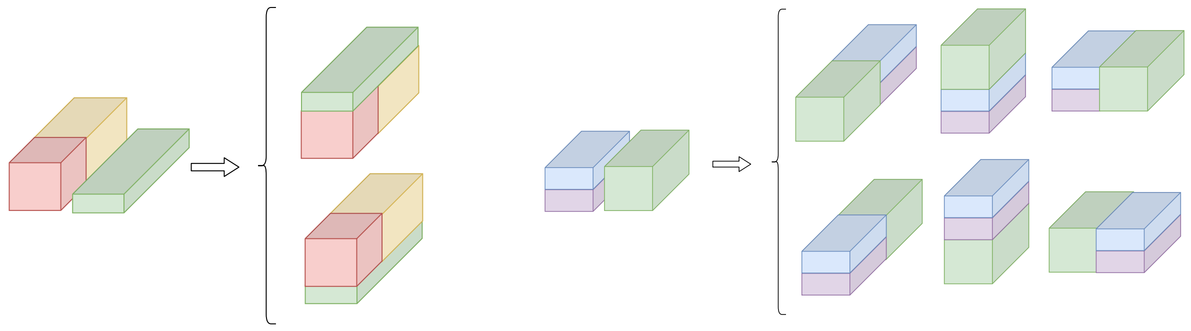

(b) Items sharing all dimensions.

Figure 7: Construction of larger blocks: first call of the preprocessing procedures.

2. *Combine items sharing all three dimensions:* two items with same length, width, and height are combined in one of the six possible blocks shown in Figure 7(b), with equal probability. Also in this case, the procedure considers all items in $\mathcal{B}$ and is then repeated iteratively.

Repeating the same combination rule more than once allows creating larger blocks made of more items, as shown in Figure 8 where each rule has been applied twice. When the procedure ends, the new set of items is ready to go through the next phases of the algorithm, starting from sorting and randomization described in Section 4.1. We do not use a tolerance factor when comparing items' dimensions, but only perfect matches, as on average this latter choice led to the best solutions.



(a) Items sharing two dimensions. (b) Items sharing all dimensions.

Figure 8: Construction of larger blocks: second call of the preprocessing procedures.

Although constructing blocks is popular in the CLP literature (see Section 1), handling blocks is more complex when dealing with the practical constraints defining our problem. Indeed, despite each item is assumed homogeneous in density, a block can be made of items with different density, which affects load balancing ($\texttt{C2}^{\text{L}}$)–($\texttt{C2}^{\text{H}}$). Moreover, even when sharing all dimensions, blocks may have different implications in terms of stability using the support factor ($\texttt{C7}$), as can be seen in Figure 8(a): In the upper block, vertical stability must be enforced to both items at the bottom of this block, whereas the lower block can be treated as a single item. Therefore, despite blocks are used to reduce space gaps and improve volume filling, as done in the literature, our application requires keeping track of the original items and their coordinates within the blocks to ensure compliance with several of the constraints. Finally, we assign the following properties to a block: (i) taxability and priority values which are the sum of the individual items' taxability and priorities, respectively, which is consistent with our loading objectives, (ii) the customer number of the later costumer to serve, (iii) the label "fragile" if at least one of the composing items is fragile, and (iv) a set of feasible orientations given by the intersection of the feasible orientation sets of the composing items.

4.3 Item sorting and controlled perturbation

The outcome from loading items sequentially in the truck depends heavily on the order in which items are considered. Thus, we aim at generating "promising" item lists that we believe would lead to good packing. After extensive experimentation based on different rules, we have decided to sort items based on decreasing taxability. Moreover, since we prefer to load items with higher priority and avoid or limit the number of unloading obstacles, we consider two additional sortings that account for these features. We describe these three different orderings below, also providing an intuition on the benefit each of them brings to the subsequent loading. At each iteration of Algorithm 1, we randomly select one of these sorting rules with equal probability.

1. Sort items by decreasing taxability π_i (ties are broken randomly). Insight: Uploading first those items with higher taxability likely produces solutions with higher objective values.

2. Sort items by decreasing priority level p_i, then sort groups with the same priority by decreasing taxability π_i (ties on priority-taxability pair are broken randomly). Insight: Solutions are more likely to be feasible (mandatory items loaded) and to include higher-priority items or blocks.

3. Sort items by decreasing customer code c_i, then sort boxes with the same c_i by decreasing taxability π_i (ties on customer-taxability pair are broken randomly). Insight: Items of the same customers are more likely to be packed closely, reducing the number of unloading obstacles.

In addition to sorting the items, at each iteration of the algorithm, we perturb the sorted list with the goal of diversifying the loading procedure. More specifically, at each iteration, we apply two probabilistic rules (one after the other) to rotate the items and perturb the loading sequence. These two rules are detailed below and each of them can be executed using one of two options, (a) or (b), which is randomly picked. Thus, at each iteration, randomization equally-likely occurs via 1(a)–2(a), 1(a)–2(b), 1(b)–2(a), or 1(b)–2(b). The presented randomization rules were defined after extensive experimentation to introduce diversification without completely disrupting the initial ordering. By running multiple RCH iterations starting from different lists, we generate different solutions that are on average of good quality, and even near-optimal in the luckiest iterations.

1. Randomize orientations:
   (a) Individual items: For each item i in the list, we randomly pick one of its feasible orientations with equal probability.
   (b) Identical items: For each subset of items sharing all dimensions, we randomly pick one of the orientations that are feasible to all items in the set with equal probability.

2. Perturb order based on similarity of consecutive items:
   (a) By volume: If $v_i/v_{i+1} \in [0.7, 1.3]$, item i swaps its position with the consecutive one in the list with probability 50%.
   (b) By weight: If $q_i/q_{i+1} \in [0.7, 1.3]$, item i swaps its position with the consecutive one in the list with probability 50%.

Sorting items before loading them sequentially is not new and is indeed a common feature of constructive methods (see, e.g., Crainic et al. 2008 where different sorting criteria are compared). Randomization has also been used in GRASP approaches (Moura and Oliveira 2005, Martínez et al. 2015, Iori et al. 2020), and is incorporated during construction when selecting the next item to load from the entire set of available items. In contrast, our approach decouples randomization and construction by relying on a *loading sequence* that is precomputed and perturbed. This leads to a much faster construction phase since each item insertion runs in $\mathcal{O}(|\mathcal{B}|)$ (see Section 4.4), while it would run in $\mathcal{O}(|\mathcal{B}|^2)$ with existing GRASP methods as all available items need to be evaluated. A randomized loading sequence is only used in one paper to our knowledge (Trivella and Pisinger 2016).

4.4 Constructive packing

Given a list of boxes (some of which may be blocks), we pack them sequentially in the truck starting from the first item in the list while complying with several constraints. To do that, we use a set of candidate locations inside the container known as *Potential Points* (henceforth PPs) from Feng et al. (2015), which are a subset of the *Extreme Points* (Crainic et al. 2008). Initially, the only two PPs available to place the first item are the corners `FLB` (front-left-bottom) and the `FRB` (front-right-bottom). After a box is packed, new PPs are generated where the following items can be loaded. The idea of using two initial PPs is to construct the packing starting from opposite sides of the truck, thus better distributing the weight along the horizontal axis to fulfill the horizontal load balancing constraint ($\texttt{C2}^{\text{H}}$) displayed in Figure 3(b). Two additional PPs at the rear of the truck, i.e., corners `RLB` (rear-left-bottom) and `RRB` (rear-right-bottom), are made available only for locating dangerous items (ADR) since these two corners are next to the unloading point.

A question that arises is how to update the set of PPs after an insertion. In Figure 9, we provide a simple example where two items are loaded starting from the left PP (i.e., corner `FLB`). When the first item is loaded, three new PPs are created whereas `FLB` is not available anymore for insertion. The second item is placed in one of the new available locations, which in turn generates three new PPs while preventing the chosen location from being selected again. The two PPs at the rear of the container are marked differently to indicate their use is limited to the placement of ADR cargo. The example in Figure 9 does not cover all possible situations. Suppose that item i is placed with its

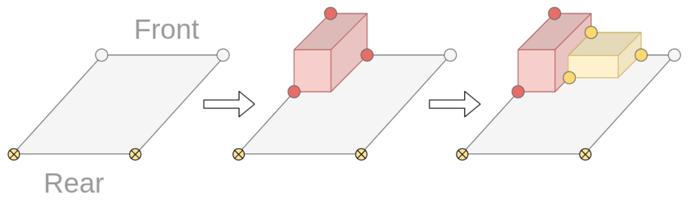


Figure 9: Example of PPs generation.

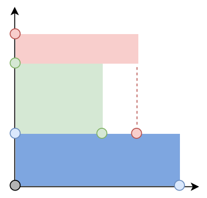


Figure 10: PP obtained by projection.

front-left-bottom corner in a position (x_i, y_i, z_i), corresponding to a PP generated by construction from the left side of the truck. Then, if item i is placed directly on the floor of the container or is stacked above another item without exceeding its support face, three new PPs will be created at coordinates $(x_i + l_i, y_i, z_i)$, $(x_i, y_i + w_i, z_i)$, and $(x_i, y_i, z_i + h_i)$, as shown indeed in Figure 9. Instead, if some vertices of box i are not supported (see, e.g., the case displayed in Figure 4), new PPs will be added by projecting the unsupported vertices onto the items below or the floor of the container. We provide a 2D example of this situation in Figure 10, where the top item is not fully supported.

The maximum number of PPs that can be created after placing an item is four, which happens when three vertices of the inserted item fall off the supported area. In this case, one PP is added at coordinates $(x_i, y_i, z_i + h_i)$ and three others result from projections. The procedure is analogous and symmetric when placing an item in a PP generated starting from the right side of the truck. Therefore, the number of PPs grows linearly with the number of loaded boxes, i.e., it is $\mathcal{O}(|\mathcal{B}|)$.

A second question is how to select the position for the next item among multiple PPs. First of all, a PP has to be feasible for placing the item, meaning that the following conditions must be satisfied: the total weight limit is not exceeded (C1), the maximum weight supported for all container zones affected by the positioning of the item is respected ($\texttt{C2}^{\text{L}}$), the insertion does not violate stackability constraints (C5), the point can accommodate ADR cargo (C6), the item is stable, i.e., it is sufficiently supported (C7), and the item does not overlap with other items nor it exceeds the boundaries of the container (standard knapsack constraints). Therefore, the choice is restricted to those PPs that jointly comply with all these conditions. Among those, we say that a PP j is preferable with respect to another s (and write $j \succ s$) if inserting the item at j implies that a larger proportion of the contact surface with the underlying item is used. Based on experiments, we find that this criterion helps increasing volume utilization and facilitates vertical stability. A graphical example of this rule is provided in Figure 11. In tie-breaking cases, we set $j \succ s$ if j has a lower x-coordinate than s, that is, we prefer loading items closer to the front of the truck.

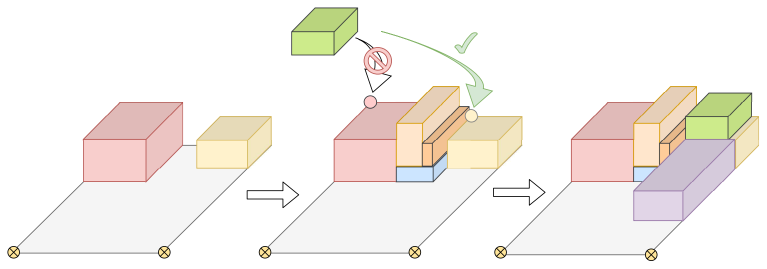


Figure 11: Illustration of PP selection rule based on contact surface.

We formalize and summarize our constructive packing procedure in Algorithm 2. The outer loop considers all items sequentially by following the order in the list $\mathcal{L}$. The inner loop considers all PPs available and keeps track of the best PP to place a given item. When an item is loaded, we remove the chosen location while adding the PPs generated by the new item. In case the insertion fails, i.e., none of the currently available PPs can accommodate the item, this item is inserted in a “retry list” $\mathcal{L}^{\text{RETRY}}$. After all items have been considered (first outer loop), we repeat the packing procedure with the retry list. The rationale is that it might be possible to pack some additional items in a second iteration as more PPs will be created and made available at that time. For example, an item may not be loadable at the first iteration because all PPs are in the front zone of the truck and adding this item in any of such locations would violate the longitudinal weight distribution (see Figure 3(a)). Nonetheless, at the second iteration involving $\mathcal{L}^{\text{RETRY}}$, new PPs might be available for placement in the other zones of the truck. To increase the chance of loading at the second iteration, we randomly pick a new orientation of items in $\mathcal{L}^{\text{RETRY}}$. Although it is possible in principle to retry insertion multiple times, we found that the benefit of extra retry attempts is usually negligible. Thus, the results we report are based on a single retry attempt. To conclude, since the number of active PPs at any time is $\mathcal{O}(|\mathcal{B}|)$ and each PP feasibility check and comparison operation is constant in time, Algorithm 2 runs in $\mathcal{O}(|\mathcal{B}|^2)$. Thus, the procedure is very quick even for large industrial instances.

Algorithm 2: Constructive packing phase of RCH

```
Input: Box list $\mathcal{L}$, empty container, potential points $\mathcal{P} = \{\texttt{BLF}, \texttt{BRF}\}$, retry list $\mathcal{L}^{\text{RETRY}} = \varnothing$
foreach item $i$ in $\mathcal{L}$ do
    Initialize best potential point $j^{\text{BEST}} = \varnothing$
    foreach potential point $j \in \mathcal{P}$ do
        if $j$ is feasible for item $i$ and $j \succ j^{\text{BEST}}$ then
            $j^{\text{BEST}} = j$
    if $j^{\text{BEST}} \neq \varnothing$ then
        Place item $i$ at location $j^{\text{BEST}}$, remove $j^{\text{BEST}}$ from $\mathcal{P}$, generate new PPs to add to $\mathcal{P}$
    else
        Insertion failed: Add item to retry list $\mathcal{L}^{\text{RETRY}}$
foreach item $i$ in $\mathcal{L}^{\text{RETRY}}$ do
    Randomly reorient item $i$ and initialize best potential point $j^{\text{BEST}} = \varnothing$
    foreach potential point $j \in \mathcal{P}$ do
        if $j$ is feasible for item $i$ and $j \succ j^{\text{BEST}}$ then
            $j^{\text{BEST}} = j$
    if $j^{\text{BEST}} \neq \varnothing$ then
        Place item $i$ at location $j^{\text{BEST}}$, remove $j^{\text{BEST}}$ from $\mathcal{P}$, generate new PPs to add to $\mathcal{P}$
Output: Loading solution $X$ including items to load, loading order, and coordinates
```

4.5 Solution evaluation

Executing Algorithm 1 results in N loading solutions, where N is potentially a large number to the order of hundreds of thousands, of which the infeasible solutions are usually below 1%. The feasible solutions are then assessed and the user receives a meaningful subset of the "best" solutions (e.g., 5 solutions) among which to choose.

As discussed in Sections 4.1 and 4.4, all loading patterns generated by RCH satisfy, by construction, the hard practical constraints. We recall that the conditions on unloading obstacles and cumulative priority are soft constraints. This means that, although our sorting phase in Section 4.1 favors solutions with a lower number of unloading obstacles and higher cumulative priority, we do not explicitly limit the number of unloading obstacles or enforce items with priorities below 10 during construction. Therefore, loading solutions are evaluated based on three criteria: total taxability, number of unloading obstacles, and cumulative priority, where the latter refers to the sum of priorities of all items that have been loaded: $\sum_{i \in \mathcal{B}} p_i \, t_i$. These three objectives may be in conflict, i.e., reducing unloading obstacles or increasing cumulative priority may require sacrificing some taxability or vice versa. When analyzing the three objectives, we can discard all dominated solutions. More precisely, we say that a solution X_a is non-dominated if there is no other solution X_b whose values of the three objectives are better (greater or lower) than or equal to the corresponding ones of X_a. As a result, the user is provided with a set S of non-dominated solutions ranked by taxability as it was shown in the example in Figure 6 (evaluation phase), where S can also be specified by the user. To facilitate the analysis, in Section 5 we compare the three objective two by time (taxability vs number of unloading obstacles and taxability vs cumulative priority), and approximate the two corresponding Pareto frontiers of our integer problem.

In addition to the three evaluation objectives, we report to the user the total weight of each packing solution, the percentage of used weight, the weight over each longitudinal zone, the volume occupied by the cargo, the percentage of volume utilization, the number of loaded items, and the coordinates of the center of mass. Using this information together with a 3D visualization tool that we have developed, the user can further explore the solution and choose the loading pattern to implement.

4.6 Upper bounds

We benchmark the cargo taxability obtained by our RCH approach against two upper (dual) bounds on the optimal taxability value based on two CLP relaxations of increasing complexity.

1. *Continuous relaxation.* We compute a greedy upper bound which neglects most constraints and assumes that fractions of items can be loaded into the container. This bound is defined as: $U^{\mathrm{C}} := \min\{U_1^{\mathrm{C}}, U_2^{\mathrm{C}}\}$, where U_1^{C} is obtained by (i) loading all mandatory items, (ii) sorting the remaining items by taxability over volume ratio, (iii) inserting sorted items until their joint volume exceeds the container's volume, (iv) if only a fraction of the item fits, consider the proportion of taxability amounting to that fraction. U_2^{C} is defined similarly but items are sorted by taxability over weight ratio and inserted until the maximum weight is reached.

2. *3D Knapsack packing*: We solve the 3DKP (1) (see Section 2.1) where we include a constraint on maximum weight $\sum_{i \in \mathcal{B}} q_i \, t_i \leqslant Q^{\text{MAX}}$, force mandatory items by fixing their variable $t_i = 1$, and inhibit for each item i its infeasible orientations by setting to zero the corresponding variables o_{ij}. This is a relaxation of our CLP that provides an actual packing where the physical knapsack constraints are enforced but load balancing, static stability, stackability, and unloading order are neglected. In our experimental analysis, the resulting mathematical formulation is solved by means of an MIP solver under a limited computational time. If the optimal solution is not found within the given time limit, the best feasible solution (if any) returned by the solver is not guaranteed to be a valid upper bound to our problem. In this case, the best upper bound available on the branch-and-cut tree of the MIP solver is returned instead.

5. Numerical study

In this section, we present our computational analysis. Before discussing instances and experiments in detail, we provide an example of how our RCH algorithm compares with EasyCargo (2020), a popular commercial loading software that the company initially tested. Specifically, we consider an industry instance and set up in EasyCargo all possible constraints handled by the software, including weight limits (C1), item orientations (C4), non-stackable items (C5), and stability (C7). However, we were not able to precisely set up our load distribution logic on longitudinal and horizontal axles (C2^{L})–(C2^{H}), loading priority (C3), ADR cargo positioning (C6), and unloading order (C8).

In Figure 12, we display an example of packing obtained by EasyCargo (left) and by RCH (right). Despite the former solution being substantially less constrained (and possibly infeasible due to mandatory items not loaded or ADR cargo incorrectly positioned), our algorithm was able to achieve a 13.5% higher taxability. The figure also shows how EasyCargo builds the solution from the left side of the truck, as opposed to our approach that uses both sides. This leads to considerable empty space on the right side of the truck and a worse load distribution on the horizontal axis.

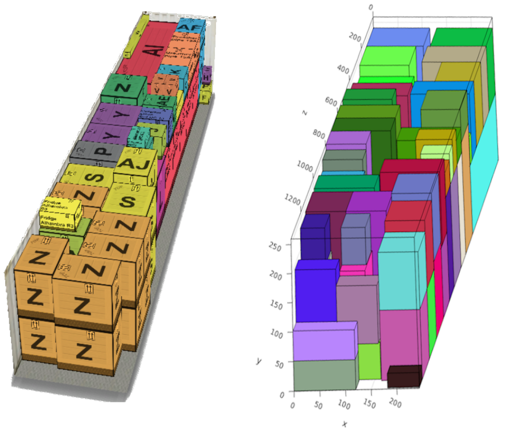


Figure 12: Solving the same instance with EasyCargo (left) and RCH (right).

In the following, we present our computational analysis. We start by describing a set of industry instances provided by ITLM Group as well as the computational setup in Section 5.1. Then, we compare the company's results with the ones obtained by RCH in Section 5.2, and perform a detailed solution analysis by showing relevant trade-offs between objectives in Section 5.3. We finally discuss the results obtained by RCH on benchmark instances for the basic CLP in Section 5.4.

5.1 Real-case instances and computational setup

We tested our RCH algorithm on 38 representative instances provided by the company, with an average of 79 boxes available for loading and a maximum of 442. All the conditions and parameter values reported below and concerning instances and constraints have been provided by the company as mandatory. In particular, items in our instances can only be rotated by 90° around the vertical axis z, i.e., boxes cannot be flipped or overturned and the base of each box must remain parallel to the floor of the container. Given the six orientations of Figure 2, rotations are hence limited to o_{i1} and o_{i2} (assuming o_{i1} is the initial item orientation). Moreover, all instances involve the same truck type, representing the most common delivery vehicle used by the company. The internal loading space has dimensions 1,360(L) $\times$ 244(W) $\times$ 260(H) cm^3 and a maximum supported weight of 26,000 kg. Longitudinally, the load must be distributed over four zones of equal length (see Figure 3(a)). Starting from the front of the truck, the supported weight cannot exceed 5,000 kg in the first zone, 6,000 kg in the second, 10,000 kg in the third, and 5,000 kg in the last. Horizontally, the acceptance zone for the center of mass extends for 1/3 of the truck's width and is centered around the midpoint (see Figure 3(b)). Regarding stability, the allowed item overhanging is 20% of its base area, i.e., 80% of the base area must be supported when stacking (see Figure 4). The constant α defining taxability is set to 300 (kg/m^3) by the transport authority.

RCH was implemented in Java and run on a laptop with an Intel Core i5 processor and 16 GB RAM. We used $N = 500$ iterations for each instance. Since RCH runs in a few seconds (usually less than 10 seconds per instance), we do not set a time limit. We compare the best solutions found by RCH with those declared by the company and with the bounds on taxability defined in Section 4.6. The knapsack packing bound is solved with Gurobi 9.1 with a time limit of 3600 seconds. Although computing this bound can be time-consuming and requires a math programming solver, note that this is just an additional tool we use to evaluate our solutions. The RCH algorithm itself is extremely fast and does not require the use of optimization solvers or the purchase of licenses by the company.

5.2 Comparison with company's solutions

For each of the 38 instances, Table 3 compares the taxability obtained from: (i) the best upper bound, (ii) the company's loading solution, and (iii) RCH. This table is organized as follows: for each instance, we report its ID, the number of boxes available for loading (column "#it"), and the number of boxes with different size/properties, which encodes the level of cargo heterogeneity (column "$\neq$"). All taxability values (columns "tax") have unit 10^3 kg. Since RCH includes randomization, we run it 10 times and provide the worst (min), the average (avg), and the best (max) taxability value achieved. The RCH gap and runtime in the last two columns refer to the average over the 10 runs. The optimality gaps are expressed as percentages with respect to the best dual bound. This is

Table 3: Comparison of taxability results.

Instance			Bound	Company			RCH				
							tax			gap (%)	time (s)
ID	#it	$\neq$	tax	tax	gap (%)	U	min	avg	max	avg	avg
1	38	28	13.6	13.6	0.0		13.6	13.6	13.6	0.0	1.2
2	32	22	23.3	12.2	47.6		23.3	23.3	23.3	0.0	1.2
3	80	54	46.5	7.9	83.0		32.0	34.5	35.0	25.8	3.6
4	135	75	60.8	37.0	39.1	U	36.1	36.2	36.3	40.5	2.8
5	77	42	31.9	13.9	56.4		23.8	25.7	27.0	19.4	5.6
6	18	18	7.3	2.6	64.4		7.3	7.3	7.3	0.0	0.2
7	85	38	34.8	34.3	1.4		34.7	34.7	34.7	0.3	2.1
8	29	23	11.1	11.1	0.0		11.1	11.1	11.1	0.0	0.3
9	155	93	45.9	31.9	30.5		27.6	28.5	29.4	38.0	6.8
10	48	31	78.0	78.0	0.0	U	78.0	78.0	78.0	0.0	0.9
11	56	15	13.7	13.6	0.7		13.7	13.7	13.7	0.0	0.7
12	44	19	17.7	17.7	0.0		17.7	17.7	17.7	0.0	0.5
13	32	17	9.2	9.2	0.0		9.2	9.2	9.2	0.0	0.5
14	148	113	59.3	53.2	10.3	U	39.3	41.0	42.7	30.9	30.1
15	129	49	49.3	46.3	6.3	U	42.8	43.2	44.1	12.4	16.8
16	70	33	40.6	35.6	12.3	U	36.7	37.5	38.2	7.8	2.2
17	27	12	16.9	16.8	0.6		16.9	16.9	16.9	0.0	0.2
18	80	37	41.3	32.0	22.5		34.8	35.3	35.8	14.5	6.7
19	27	9	24.6	24.6	0.0		24.6	24.6	24.6	0.0	0.2
20	11	7	6.5	6.5	0		6.5	6.5	6.5	0.0	0.1
21	41	26	24.5	24.4	0.4		23.8	23.9	24.5	2.7	0.5
22	102	67	51.1	21.4	58.1		32.2	35.9	38.1	29.6	10.7
23	24	3	23.7	22.2	6.3		23.7	23.7	23.7	0.0	0.1
24	209	112	57.0	11.2	80.4		38.7	40.9	42.7	28.2	16.8
25	79	33	26.8	24.6	8.2		24.4	25.1	26.2	6.4	3.3
26	2	2	1.6	1.6	0.0		1.6	1.6	1.6	0.0	0.0
27	20	11	4.1	4.0	2.4		4.1	4.1	4.1	0.0	0.1
28	11	6	2.6	2.3	11.5		2.6	2.6	2.6	0.0	0.1
29	58	26	36.1	52.5	−45.4	U	35.2	35.2	35.2	2.5	1.6
30	22	4	25.2	25.2	0.0		25.2	25.2	25.2	0.0	0.1
31	442	20	25.0	12.1	51.6		19.5	20.7	21.8	17.2	30.4
32	126	20	37.2	12.3	67.0		23.5	24.3	25.7	34.6	20.0
33	10	1	10.4	10.4	0.0		10.4	10.4	10.4	0.0	0.0
34	67	36	46.1	44.9	2.6		33.4	35.6	36.5	22.8	0.9
35	153	83	73.3	52.0	29.1		53.7	55.7	57.3	24.2	5.9
36	64	42	37.2	41.0	−10.0	U	36.8	36.8	36.8	0.9	2.8
37	183	56	40.1	41.1	−1.3	U	39.1	39.2	39.7	3.5	5.3
38	71	37	40.5	41.1	−1.3	U	35.3	35.4	35.6	12.7	3.8
with U	79.1	34.7	31.3	24.8	16.7		26.1	26.7	27.2	9.9	4.8
w/o U	74.5	31.1	26.2	17.8	21.5		21.2	21.8	22.3	9.1	4.1

given by the knapsack packing model (mathematically, it is tighter than the continuous relaxation) with the exception of the largest instances for which GUROBI was not able to return any bound within 3600 seconds and hence the continuous relaxation is considered. We also report two different average values in the last rows, with the first average being over all instances while the second excluding instances marked with "U", for which the company's solution is provably unfeasible.

From the table, we notice that: (i) RCH achieves higher taxability in 18 instances out of 38 instances while the company leads in 10 instances, and (ii) the average taxability improvement from RCH is 7.7% (considering the average run), with 6.8% lower optimality gap on average. To understand these results, it is important to mention that we are not provided with the loading coordinates of boxes in the company's solutions as they do not currently store such information: we only know which items have been loaded and which have not been. Consequently, we cannot determine whether all constraints (`C1`)–(`C8`) have been fulfilled. Nevertheless, by computing the total cargo weight we can state that 9 solutions do not comply with the weight limits. These solutions are infeasible according to our specifications and we mark them with "U" in the table. If we exclude these "obviously infeasible" solutions from the average, the improvement from RCH is 22.4% (with 12.4% lower optimality gap), and the company only leads in 3 solutions. We suspect (but we are unable to verify this claim) that many other solutions do not satisfy positioning and weight distribution constraints; excluding them would make the improvement for RCH even larger.

These results on taxability are highly meaningful for the company as they reveal significant potential for increasing profits associated with shipping, as well as, provide a direct way to tap this potential through RCH. In addition to taxability improvements, by better complying with the constraints the company can avoid fines due to excessive weight or incorrect load distribution, increase safety during transportation, and save on fuel consumption.

The RCH runtime for 500 iterations is about 5 seconds on average and is below 10 seconds in all but six instances. This means that RCH is suitable for practical use and can produce solutions in near real-time, i.e., on-demand. In instances number 14 and 31, the runtime is slighly higher and reaches a maximum of 30 seconds. To further reduce waiting time for the user, one could dynamically evaluate the set of currently available solutions every 5 seconds so that the user is quickly provided with some loading patterns to visualize. As soon as better solutions are found, they can be provided as well. Alternatively, one could parallelize the RCH execution over iterations.

5.3 Analysis of alternative loading objectives

In this section, we consider one of our industry instances and examine in more detail some aspects of the solution process and the trade-offs that arise between different loading objectives.

First, we show how randomization contributes to the solution quality by illustrating its effect in Figure 13. The dashed line represents the taxability achieved by packing a (non-randomized) list where items are sorted by non-increasing taxability. Moreover, the dots represent the taxability values obtained during 1000 randomized iterations. Packing the non-randomized list results in a taxability value (dashed line) that is visibly higher than the average of randomized lists (dots), which is intuitive since sorting by non-increasing taxability is consistent with the objective to maximize taxability. Although perturbing this order worsens results on average, in some iterations the taxability improves. In this instance, the best taxability value is found at iteration 862, showing 16% increase with respect to the initial sorting. This analysis confirms that randomization is extremely helpful to create diversification and to discover high-value packing configurations. In this particular run of the algorithm, the best solution is found in one of the latest iterations, but already within the first 200 iterations we can find a solution that looses from the best one only by 0.2%.

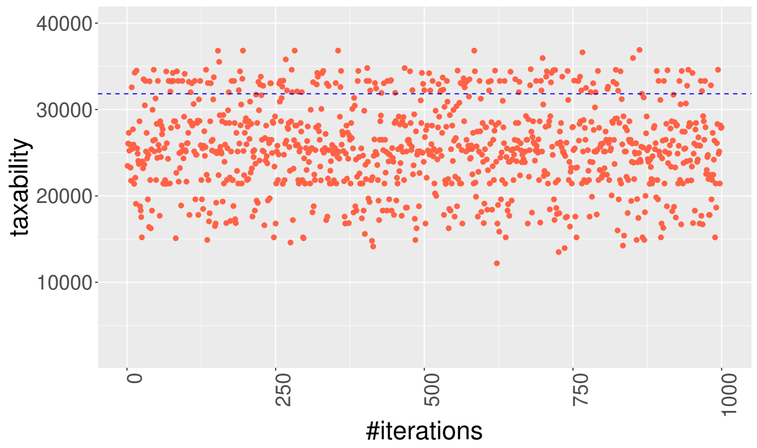


Figure 13: Impact of randomization on taxability.

Our analysis has so far been centered around taxability. As discussed in Section 4.5, the number of unloading obstacles and the cumulative priority are other important aspects to consider when evaluating a solution. In Figure 14, we display the trade-off between taxability and the number of unloading obstacles. Each dot in this figure represents the taxability-obstacle pair achieved in a randomized iteration. The solid line is an approximation of the Pareto frontier connecting non-dominated solutions, which reveals that higher taxability levels can be obtained at the cost of a higher number of unloading obstacles. In this example, the loading solution with the highest taxability (precisely 36,900) involves as many as 29 unloading obstacles. As the solutions become more constrained (i.e., a lower number of unloading obstacles is allowed) the taxability decreases and the best obstacle-free taxability is only 18,490, which is about half of the overall-best taxability. Such a conflicting relation between these two loading objectives can be explained by the fact that the higher-taxability solutions are usually those where a larger number of items is loaded, which in turn makes it easier for more unloading obstacles to be generated.

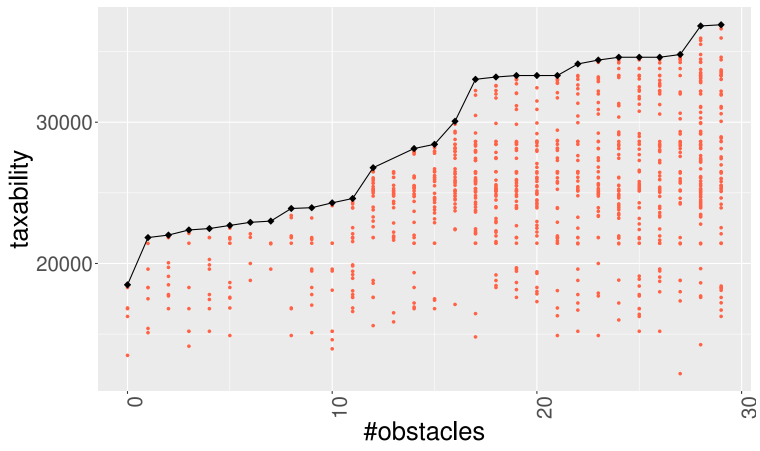


Figure 14: Approximation of the taxability-obstacle Pareto frontier.

The spread of the approximated Pareto frontier in both unloading obstacles and taxability dimensions underscores how critical it is to account for this trade-off. Some portions of the frontier are almost flat, suggesting that a significant reduction of unloading obstacles can be achieved for a relatively small decrease in taxability. For example, reducing unloading obstacles from 21 to 17 causes a loss in taxability by only 0.8%. Choosing a solution in this frontier is non-trivial, but quantifying the economic impact of unloading obstacles (e.g., depending on the box's dimensions, weight and the estimated time to move it) may help in this decision as we discuss in Section 6.

Finally, Figure 15 illustrates the relation that exists between taxability and cumulative priority. In contrast to the former taxability-obstacle trade-off, the figure shows that taxability and cumulative priority are positively correlated. In other words, we expect that the higher the taxability, the higher also is the cumulative priority, which can also be explained by the higher-taxability solutions typically loading more items, and hence more easily reaching a high cumulative priority count. In conclusion, managing cumulative priority as an additional objective appears less critical.

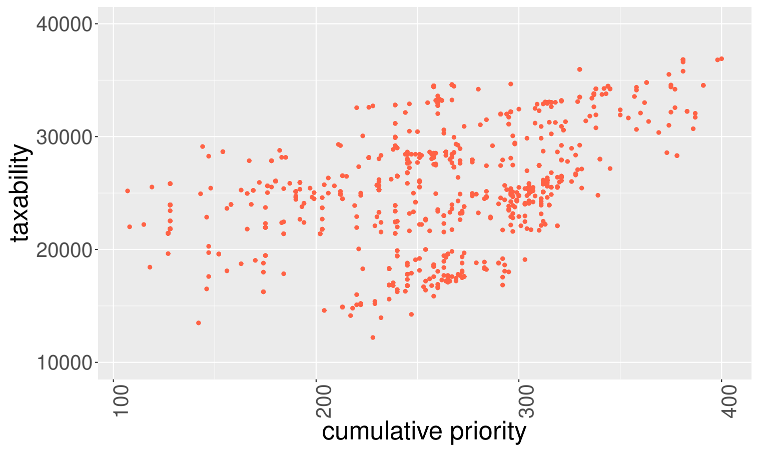


Figure 15: Taxability versus cumulative priority.

5.4 Additional tests on benchmark instances

To complete our analysis, we tested our method on the "BR" benchmark instances (Bischoff and Ratcliff 1995, Davies and Bischoff 1999), which consist of 15 instance sets (BR1–BR15) varying by the level of heterogeneity of the boxes and each comprising 100 instances. The BR instances are commonly used in the literature for cargo loading problems maximizing the total volume of loaded boxes but exclude almost all the real-life constraints studied in this paper. The only constraint we consider is full-support stability since benchmark results from the literature accounting for this constraint are available. This is a special case of the stability constraints (C7) where a generic support factor can be specified. Moreover, we set the taxability of each box equal to its volume.

Running RCH for 30 seconds per instance resulted in an average volume utilization in the range of 83–90% depending on the instance set, with an average across all 1500 instances of 86.5%. On one hand, this average is 6.5% below the best-known results on the BR instances based on the same running time of 30 seconds, which are those achieved by the tree-search-based method of Araya et al. (2017). This is expected as our framework has been developed for a different problem and is able to comply with many practical constraints ignored by this simplified formulation. On the other hand, even on such basic CLP instances, RCH outperforms other constructive methods that are more similar in structure. For example, our average volume occupation is 3.5% higher than that of the GRAPS by Martínez et al. (2015), which is a significant improvement considering that we do not use an improvement phase due to the computational requirements set by the company. Finally, we noticed that almost identical results are obtained by RCH when setting the running time to much lower limits, e.g., 10 seconds, which makes our results reliable under strict time requirements.

6. Conclusion

Motivated by a logistics company, we studied a real CLP problem that embeds numerous practical constraints (weight limits, weight distribution and load balancing, loading priorities, orientations, stacking, dangerous items, stability, multi-drop shipments), some of which are new to the extant literature such as treating the the number of unloading obstacles as soft constraints. Due to the complexity of the problem and large-scale industry instances to solve, we have developed a novel randomized constructive heuristic algorithm (RCH) that can produce multiple packing solutions in a few seconds. On real instances, our RCH solutions outperform the industry solutions significantly in terms of taxability while accounting for practical constraints and relevant trade-offs between objectives. We also introduce dual bounds based on CLP relaxations to establish optimality gaps.

Our collaborating company has decided to integrate RCH in its day-to-day loading process. This strategy can help the company improving at the same time its financial performance (by increasing profits and reducing time and effort when loading trucks) and its operational sustainability (reducing the number of shipments decreases fuel consumption and the impact on the environment). The resulting annual savings are estimated at up to one million Euros and one thousand tons of CO_2. The tool has been also positively welcomed by the employees responsible for loading, in particular, because it does not impose a one-and-only solution to implement. Instead, by providing a few different loading patterns and their 3D visualizations, the warehouse staff can choose the most suitable one by considering taxability, unloading obstacles, priorities, easiness to load, etc. Therefore, RCH is not meant to replace the expertise of the employees; on the contrary, it is a decision support tool that can help to combine this expertise with the potential of optimization techniques.

This work paves the way both for further research and for the integration of Operations Research methods in other processes part of the company's workflow. Providing a more accurate representation of the unloading obstacles is a possible future development. This could be done by introducing a penalty that accounts for the real impact, in terms of time and cost, of dealing with an obstacle, and that could be a function of the box's dimensions, weight, and type of relocation move. Another avenue of future research concerns the integration of the proposed methodology with a revenue management system to help the managerial team decide how much to charge shipping orders, by anticipating the impact that accepting orders brings when optimizing the load of a truck.

Acknowledgments

We thank Paolo Giannotti, Maria Antonietta Spera, Federico Pozzi Chiesa, and the rest of the team at ITLM Group involved in the project for supporting this collaboration with knowledge and data. We also thank the review team for constructive feedback that significantly improved the manuscript.

References

M. Alonso, R. Alvarez-Valdes, M. Iori, F. Parreño, and J. Tamarit. Mathematical models for multicontainer loading problems. *Omega*, 66:106–117, 2017. doi: 10.1016/j.cor.2017.01.012.

M. Alonso, R. Alvarez-Valdes, M. Iori, and F. Parreno. Mathematical models for multi-container loading problems with practical constraints. *Computers & Industrial Engineering*, 127:722–733, 2019. doi: 10.1016/j.cie.2018.11.012.

R. R. Amossen and D. Pisinger. Multi-dimensional bin packing problems with guillotine constraints. *Computers & Operations Research*, 37(11):1999–2006, 2010. doi: 10.1016/j.cor.2010.01.017.

I. Araya and M.-C. Riff. A beam search approach to the container loading problem. *Computers & Operations Research*, 43:100–107, 2014. doi: 10.1016/j.cor.2013.09.003.

I. Araya, K. Guerrero, and E. & Nunez. Vcs: A new heuristic function for selecting boxes in the single container loading problem. *Computers & Operations Research*, 82:27–35, 2017. doi: 10.1016/j.cor.2017.01.002.

M. M. Baldi, G. Perboli, and R. Tadei. The three-dimensional knapsack problem with balancing constraints. *Applied Mathematics and Computation*, 218(19):9802–9818, 2012. doi: 10.1016/j.amc.2012.03.052.

E. Bischoff. Three-dimensional packing of items with limited load bearing strength. *European Journal of Operational Research*, 168(3):952–966, 2006. doi: 10.1016/j.ejor.2004.04.037.

E. E. Bischoff. Stability aspects of pallet loading. *OR Spektrum*, 13(4):189–197, 1991. doi: 10.1007/BF01719394.

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

A. Bortfeldt and H. Gehring. A hybrid genetic algorithm for the container loading problem. *European Journal of Operational Research*, 131(1):143–161, 2001. doi: 10.1016/S0377-2217(00)00055-2.

A. Bortfeldt and G. Wäscher. Constraints in container loading–a state-of-the-art review. *European Journal of Operational Research*, 229(1):1–20, 2013. doi: 10.1016/j.ejor.2012.12.006.

A. Bortfeldt, H. Gehring, and D. Mack. A parallel tabu search algorithm for solving the container loading problem. *Parallel Computing*, 29(5):641–662, 2003. doi: 10.1016/S0167-8191(03)00047-4.

M. Colombi, Á. Corberán, R. Mansini, I. Plana, and J. M. Sanchis. The directed profitable rural postman problem with incompatibility constraints. *European Journal of Operational Research*, 261(2):549–562, 2017. doi: 10.1016/j.ejor.2017.02.002.

T. G. Crainic, G. Perboli, and R. Tadei. Extreme point-based heuristics for three-dimensional bin packing. *INFORMS Journal on Computing*, 20(3):368–384, 2008. doi: 10.1287/ijoc.1070.0250.

E. da Silva, A. Leão, F. Toledo, and T. Wauters. A matheuristic framework for the three-dimensional single large object placement problem with practical constraints. *Computers & Operations Research*, 124: 105058, 2020. doi: 10.1016/j.cor.2020.105058.

A. P. Davies and E. E. Bischoff. Weight distribution considerations in container loading. *European Journal of Operational Research*, 114(3):509–527, 1999. doi: 10.1016/S0377-2217(98)00139-8.

EasyCargo. Easycargo, 2020. URL https://www.easycargo3d.com/. Accessed 2 October 2020.

J. Egeblad and D. Pisinger. Heuristic approaches for the two-and three-dimensional knapsack packing problem. *Computers & Operations Research*, 36(4):1026–1049, 2009. doi: 10.1016/j.cor.2007.12.004.

J. Egeblad, C. Garavelli, S. Lisi, and D. Pisinger. Heuristics for container loading of furniture. *European Journal of Operational Research*, 200(3):881–892, 2010. doi: 10.1016/j.ejor.2009.01.048.

M. Eley. A bottleneck assignment approach to the multiple container loading problem. *OR Spectrum*, 25(1): 45–60, 2003. doi: 10.1007/s002910200113.

T. Fanslau and A. Bortfeldt. A tree search algorithm for solving the container loading problem. *INFORMS Journal on Computing*, 22(2):222–235, 2010. doi: 10.1287/ijoc.1090.0338.

G. Fasano. A mip approach for some practical packing problems: Balancing constraints and tetris-like items. *Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, 2(2):161–174, 2004. doi: 10.1007/s10288-004-0037-7.

X. Feng, I. Moon, and J. Shin. Hybrid genetic algorithms for the three-dimensional multiple container packing problem. *Flexible Services and Manufacturing Journal*, 27(2-3):451–477, 2015. doi: 10.1007/s10696-013-9181-8.

H. Gehring and A. Bortfeldt. A genetic algorithm for solving the container loading problem. *International Transactions in Operational Research*, 4(5-6):401–418, 1997. doi: 10.1111/j.1475-3995.1997.tb00095.x.

M. Gendreau, M. Iori, G. Laporte, and S. Martello. A tabu search algorithm for a routing and container loading problem. *Transportation Science*, 40(3):342–350, 2006. doi: 10.1287/trsc.1050.0145.

J. A. George and D. F. Robinson. A heuristic for packing boxes into a container. *Computers & Operations Research*, 7(3):147–156, 1980. doi: 10.1016/0305-0548(80)90001-5.

M. Hifi, I. Kacem, S. Nègre, and L. Wu. A linear programming approach for the three-dimensional bin-packing problem. *Electronic Notes in Discrete Mathematics*, 36:993–1000, 2010. doi: 10.1016/j.endm.2010.05.126.

M. Iori, M. Locatelli, M. C. Moreira, and T. Silveira. Reactive GRASP-based algorithm for pallet building problem with visibility and contiguity constraints. In *International Conference on Computational Logistics*, pages 651–665. Springer, 2020. doi: 10.1007/978-3-030-59747-4_42.

ITLM. ITLM Group home page, 2021. URL https://itlmgroup.com/index.html. Accessed 1 June 2021.

T. Jamrus and C.-F. Chien. Extended priority-based hybrid genetic algorithm for the less-than-container loading problem. *Computers & Industrial Engineering*, 96:227–236, 2016. doi: 10.1016/j.cie.2016.03.030.

L. Junqueira, R. Morabito, and D. S. Yamashita. Three-dimensional container loading models with cargo stability and load bearing constraints. *Computers & Operations Research*, 39(1):74–85, 2012. doi: 10.1016/j.cor.2010.07.017.

Kearney. CSCMP's 31st annual state of logistics report, 2020. URL `https://cscmp.org/CSCMP/Educate/State_of_Logistics_2020.aspx`. Accessed 2 October 2020.

D. V. Kurpel, C. T. Scarpin, J. E. P. Junior, C. M. Schenekemberg, and L. C. Coelho. The exact solutions of several types of container loading problems. *European Journal of Operational Research*, 284(1):87–107, 2020. doi: 10.1016/j.ejor.2019.12.012.

A. Lim, B. Rodrigues, and Y. Wang. A multi-faced buildup algorithm for three-dimensional packing problems. *Omega*, 31(6):471–481, 2003. doi: 10.1016/j.omega.2003.08.004.

A. Lim, H. Ma, C. Qiu, and W. Zhu. The single container loading problem with axle weight constraints. *International Journal of Production Economics*, 144(1):358–369, 2013. doi: 10.1016/j.ijpe.2013.03.001.

S. Martello, D. Pisinger, and D. Vigo. The three-dimensional bin packing problem. *Operations Research*, 48 (2):256–267, 2000. doi: 10.1287/opre.48.2.256.12386.

D. A. Martínez, R. Alvarez-Valdes, and F. Parreño. A grasp algorithm for the container loading problem with multi-drop constraints. *Pesquisa Operacional*, 35(1):1–24, 2015. doi: 10.1590/0101-7438.2015.035.01.0001.

I. Moon and T. V. L. Nguyen. Container packing problem with balance constraints. *OR Spectrum*, 36(4): 837–878, 2014. doi: 10.1007/s00291-013-0356-1.

A. Moura and J. F. Oliveira. A GRASP approach to the container-loading problem. *IEEE Intelligent Systems*, 20(4):50–57, 2005. doi: 10.1109/MIS.2005.57.

O. X. Nascimento, T. A. Queiroz, and L. Junqueira. Practical constraints in the container loading problem: Comprehensive formulations and exact algorithm. *Computers & Operations Research*, 128:105186, 2021. doi: 10.1016/j.cor.2020.105186.

V. E. Ocloo, A. R. Fügenschuh, and O. M. Pamen. A new mathematical model for a 3d container packing problem. *Cottbus Mathematical Preprints COMP 12(2020)*, 2020. doi: 10.26127/btuopen-5088.

L. d. A. Oliveira, V. L. de Lima, T. A. de Queiroz, and F. K. Miyazawa. The container loading problem with cargo stability: a study on support factors, mechanical equilibrium and grids. *Engineering Optimization*, pages 1–20, 2020. doi: 10.1080/0305215X.2020.1779250.

C. Paquay, M. Schyns, and S. Limbourg. A mixed integer programming formulation for the three-dimensional bin packing problem deriving from an air cargo application. *International Transactions in Operational Research*, 23(1-2):187–213, 2016. doi: 10.1111/itor.12111.

F. Parreño, R. Alvarez-Valdés, J. M. Tamarit, and J. F. Oliveira. A maximal-space algorithm for the container loading problem. *INFORMS Journal on Computing*, 20(3):412–422, 2008. doi: 10.1287/ijoc.1070.0254.

D. Pisinger. Heuristics for the container loading problem. *European Journal of Operational Research*, 141 (2):382–392, 2002. doi: 10.1016/S0377-2217(02)00132-7.

PitneyBowes. Pitneybowes, 2020. URL `https://www.pitneybowes.com`. Accessed 2 October 2020.

H. Pollaris, K. Braekers, A. Caris, G. Janssens, and S. Limbourg. Vehicle routing problems with loading constraints: state-of-the-art and future directions. *OR Spectrum*, 37:297–330, 2015. doi: 10.1007/s00291-014-0386-3.

A. G. Ramos, E. Silva, and J. F. Oliveira. A new load balance methodology for container loading problem in road transportation. *European Journal of Operational Research*, 266(3):1140–1152, 2018. doi: 10.1016/j.ejor.2017.10.050.

SDC. Shipment stats continue to grow during pandemic, 2020. URL `https://www.sdcexec.com/transportation/press-release/21129418/consignor-shipment-stats-continue-to-grow-during-pandemic`. Accessed 2 October 2020.

E. Silva, A. G. Ramos, and J. F. Oliveira. Load balance recovery for multi-drop distribution problems: A mixed integer linear programming approach. *Transportation Research Part B: Methodological*, 116: 62–75, 2018. doi: 10.1016/j.trb.2018.08.001.

E. F. Silva, T. A. M. Toffolo, and T. Wauters. Exact methods for three-dimensional cutting and packing: A comparative study concerning single container problems. *Computers & Operations Research*, 109: 12–27, 2019. doi: 10.1016/j.cor.2019.04.020.

Statista. Global parcel shipping volume between 2013 and 2026, 2020. URL https://www.statista.com/statistics/1139910/parcel-shipping-volume-worldwide/. Accessed 2 October 2020.

C. D. Tarantilis, E. E. Zachariadis, and C. T. Kiranoudis. 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, 2009. doi: 10.1109/TITS.2009.2020187.

A. Trivella and D. Pisinger. The load-balanced multi-dimensional bin-packing problem. *Computers & Operations Research*, 74:152–164, 2016. doi: 10.1016/j.cor.2016.04.020.

J.-F. Tsai, P.-C. Wang, and M.-H. Lin. A global optimization approach for solving three-dimensional open dimension rectangular packing problems. *Optimization*, 64(12):2601–2618, 2015. doi: 10.1080/02331934.2013.877906.

L. Wang, S. Guo, S. Chen, W. Zhu, and A. Lim. Two natural heuristics for 3d packing with practical loading constraints. In *Pacific Rim International Conference on Artificial Intelligence*, pages 256–267. Springer, 2010. doi: 10.1007/978-3-642-15246-7_25.

G. Wäscher, H. Haußner, and H. Schumann. An improved typology of cutting and packing problems. *European journal of operational research*, 183(3):1109–1130, 2007. doi: 10.1016/j.ejor.2005.12.047.

WEF. The future of the last-mile ecosystem, 2020. URL http://www3.weforum.org/docs/WEF_Future_of_the_last_mile_ecosystem.pdf. Accessed 2 October 2020.

X. Zhao, J. A. Bennell, T. Bektaş, and K. Dowsland. A comparative review of 3d container loading algorithms. *International Transactions in Operational Research*, 23(1-2):287–320, 2016. doi: 10.1111/itor.12094.