European Journal of Operational Research xxx (xxxx) xxx

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Innovative Applications of O.R.

Modeling soft unloading constraints in the multi-drop container loading problem

Guillem Bonet Filella[a], Alessio Trivella[b,*], Francesco Corman[a]

[a] Institute for Transport Planning and Systems, ETH Zürich, Zürich 8092, Switzerland
[b] Industrial Engineering and Business Information Systems, University of Twente, AE Enschede 7500, the Netherlands

ARTICLE INFO

Article history:
Received 24 September 2021
Accepted 19 October 2022
Available online xxx

Keywords:
Packing
Container loading
Multi-drop shipments
Mixed-integer programming
Improvement heuristic

ABSTRACT

The multi-drop container loading problem (MDCLP) requires loading a truck so that boxes can be unloaded at each drop-off point without rearranging other boxes to deliver later. However, modeling such unloading constraints as hard constraints, as done in the literature, limits the flexibility to optimize the packing and utilize the vehicle capacity. We instead propose a more general approach that considers soft unloading constraints. Specifically, we penalize unnecessary relocations of boxes using penalty functions that depend on the volume and weight of the boxes to move as well as the type of move. Our goal is to maximize the value of the loaded cargo including penalties due to violations of the unloading constraints. We provide a mixed-integer linear programming formulation for the MDCLP with soft unloading constraints, which can solve to optimality small-scale instances but is intractable for larger ones. We thus propose a heuristic framework based on a randomized extreme-point constructive phase and a subsequent improvement phase. The latter phase iteratively destroys regions in the packing space where high penalties originate, and reconstructs them. Extensive numerical experiments involving different instances and penalties highlight the advantages of our method compared to a commercial optimization solver and a heuristic from the literature developed for a related problem. They also show that our approach significantly outperforms: (i) the hard unloading constraints approach, and (ii) a sequential heuristic that neglects unloading constraints first and evaluates the penalties afterwards. Our findings underscore the relevance of accounting for soft unloading constraints in the MDCLP.

1. Introduction

The operations of logistics systems and the efficiency in delivering goods play a crucial role in a world where goods and services must be quickly accessible. With the increasing worldwide trends toward globalization, digitalization, and urbanization, the logistics sector has been experiencing an unprecedented growth in recent years. For example, the global logistics market was valued at 10.3 billion USD in 2017 and the compound annual growth rate from 2017 to 2023 was estimated at about 3.5%, resulting in an expected market value of 12.9 billion USD by 2023 (Research & Markets, 2018). Nevertheless, this rapid growth is coupled with enormous challenges and the need to improve and update methods and technologies used in logistics and transportation systems.

Just the last year, the Covid-19 pandemic proved to be a major challenge for logistics operators around the world. Whether because of the growth in online retail and home delivery or the global distribution of vaccines, many businesses within logistics and delivery have been put under test. For instance, in April 2020, the Swiss Post found itself overwhelmed by the increased demand caused by the first lockdown and was forced to introduce daily quotas on the number of parcels, which affected more than 100 corporate clients (Post, 2020b). Preventing a parcel collapse in the following months required increasing delivery capacity as well as introducing analytics to forecast demand and optimize operations (Post, 2020a). This case exemplifies the need for logistics companies to increase their performance to cope with demand, which regardless of Covid-19 is expected to increase substantially (see, e.g., Statista, 2020), and be prepared for future challenges in general.

While several problems arising in logistics have been tackled and optimized using operations research techniques for decades, such as the container loading problem (CLP; Pisinger, 2002) and

* Corresponding author.
E-mail address: a.trivella@utwente.nl (A. Trivella).

https://doi.org/10.1016/j.ejor.2022.10.033

the vehicle routing problem (VRP; Laporte, 2009), some practically-relevant variants of these problems have been less studied and still embed significant potential for improvement. In this paper, we consider the multi-drop CLP (henceforth MDCLP), which includes *unloading constraints*. Specifically, this problem deals with optimally loading a vehicle for delivering cargo to multiple customers while accounting for the undesired relocation of boxes during the subsequent delivery process. Since the delivery order is assumed fixed, the routing component is not relevant here. Unloading constraints have been studied when incorporating practical constraints in the CLP (Nascimento, Queiroz, & Junqueira, 2021) and loading constraints in the VRP (Pollaris, Braekers, Caris, Janssens, & Limbourg, 2015). However, the vast majority of works in these areas have either neglected such constraints, which in practice leads to time consuming (i.e., costly) cargo relocations during unloading, or completely forbidden relocations of items during the delivery process, resulting in more constrained loading solutions that carry less cargo volume or value.

This study takes an intermediate perspective that treats unloading constraints in the MDCLP as soft constraints. We do this by means of penalty functions that punish relocations in a flexible manner based on properties of the items to move (e.g., weight and size) as well as the type of relocation. We propose a mathematical programming formulation that explicitly accounts for soft unloading constraints and that reduces to a mixed-integer liner program (MILP) under linear penalties. As this model is intractable for large instances, we further propose a heuristic loading framework based on construction and improvement phases. The latter phase identifies promising regions in a packing solution where many conflicts occur, and reconstructs them, also incorporating some randomization.

We performed a numerical study involving a known set of 1500 instances (Bischoff, Janetz, & Ratcliff, 1995; Davies & Bischoff, 1999), 4 penalty function specifications, and 3 region reconstruction techniques. We found that considering soft unloading constraints in the MDCLP can lead to significantly higher objective values (i.e., cargo value minus penalties) compared to: (i) the hard unloading constraints approach, by up to 12%, and (ii) a sequential approach that neglects the unloading constraints when loading boxes and assesses penalties a posteriori, by up to 15%. This significant value underscores the shortcoming of the common but inflexible models relying on hard constraints. We also show that our new method is needed because: (i) an MDCLP heuristic from the literature that simply counts the number of relocations but does not minimize the penalties may be inefficient, and (ii) off-the-shelf commercial optimization solvers can only solve to optimality tiny instances. The optimization methods we develop and related findings thus help meeting a real and unresolved need of logistics operators to handle unloading constraints in a flexible and practical manner (Gajda, Trivella, Mansini, & Pisinger, 2022).

The rest of this paper is organized as follows. Section 2 reviews the related literature. Section 3 formally defines the MDCLP with soft unloading constraints and presents the mathematical programming formulation with an illustrative example. The heuristic method is described in Section 4 and tested in a detailed numerical study in Section 5. We conclude in Section 6 and provide future research directions.

2. Literature review

We review essential literature related to the CLP in Section 2.1 and then focus on the unloading constraints in Section 2.2. We summarize the contributions of our work in Section 2.3.

2.1. Container loading problem

Given a set of three-dimensional rectangular-shaped items/boxes, each provided with a value, the CLP aims at selecting a subset of these items and loading them into a larger rectangular-shaped container so that the value of the loaded items is maximized. The CLP is a well-studied problem in operations research and is also known as the *three-dimensional (single orthogonal) knapsack problem* according to the typology of cutting and packing problems by Wäscher, Haußner, & Schumann (2007).

To tackle the CLP, both exact and heuristic methods have been developed, with the latter being significantly more popular as the problem is very hard to solve to optimality even for a few dozen items (Silva, Toffolo, & Wauters, 2019). Thus, exact methods are still far from meeting the requirements of the modern logistics industry where large instances must be solved within seconds (Gajda et al., 2022). In our paper, we propose both a mathematical program that can solve small instances optimally, illustrating the usefulness of soft unloading constraints, and a heuristic able to deal with large instances.

Heuristic approaches for the CLP have recently been classified by Zhao, Bennell, Bektaş, & Dowsland (2016), that distinguish between *construction* and *improvement* methods. While the former methods build a packing solution from scratch by loading items starting from an empty container, the latter are designed to improve an existing solution. In our heuristic we use both construction and improvement methods.

Commonly used construction heuristics are based on wall-building or layer-building schemes. The wall-building scheme, first proposed by George & Robinson (1980), creates virtual walls in the container and considers the detached spaces as smaller packing problems. Similar to the previous, the layer-building scheme creates layers that must be completely filled by items of comparable height before a new layer can be initiated (Bischoff et al., 1995; Terno, Scheithauer, Sommerweiß, & Riehme, 2000). The described approaches work well for weakly-heterogeneous instances with only a few distinct item types, i.e., made of few groups of identical items. Another set of algorithms use the maximal-space strategy where the placement of items is guided by information on the empty spaces in the container. First proposed by Lai, Xue, & Xu (1998), this approach was further developed by Parreño, Alvarez-Valdés, Tamarit, & Oliveira (2008) combining it with a greedy randomized adaptive search procedure (GRASP). Martello, Pisinger, & Vigo (2000) established the concept of *corner points*, which are promising locations for the sequential placement of items that are updated after each insertion. The concept of corner points is further extended by Crainic, Perboli, & Tadei (2008) by introducing the *extreme points*, which also use projections to generate new points. These "provide the means to exploit the free space defined inside a packing by the shapes of the items already in the container". Our construction phase makes use of an extreme-point heuristic. Finally, more complex tree-search based methods use a decision tree of limited depth to evaluate each insertion and were also proven effective (Araya, Guerrero, & &. Nunez, 2017; Fanslau & Bortfeldt, 2010).

Improvement techniques are diverse, of which we state a few. GRASP algorithms for the CLP embed an improvement phase that is often based on local search (Iori, Locatelli, Moreira, & Silveira, 2020; Moura & Oliveira, 2005). Trivella & Pisinger (2016) use the interval graph representation of multi-dimensional packing by Fekete, Schepers, & Van der Veen (2007) to develop a local search framework operating on graphs. Parreño, Alvarez-Valdes, Oliveira, & Tamarit (2010) consider five different improvement steps in a variable neighborhood search (VNS) algorithm, such as the swap of items or the re-packing of regions in the container. Our improvement phase is also based on region reconstruction but employs

new criteria for the region selection and reconstruction, which are designed to reduce penalties due to violations of the unloading constraints.

The literature in which practical constraints such as load balancing and stability are incorporated in packing problems is vast and we do not attempt to review it (e.g., Trivella & Pisinger, 2017, Alonso, Alvarez-Valdes, Iori, & Parreno, 2019, Gajda et al., 2022, Nascimento et al., 2021). For a comprehensive review of such constraints see Bortfeldt & Wäscher (2013). Since multi-drop situations are common in practice, considering the MDCLP and unloading constraints is important and we discuss related works next.

2.2. Unloading constraints

The unloading constraints are relevant for multi-drop deliveries, i.e., when a container or delivery vehicle carries items for multiple customers to serve at different locations. In these situations, it is important to avoid relocating items during the sequential unloading operations as this can be time consuming and/or complex to perform at a customer's venue. In other words, the items for the customer currently being served must be easily unloaded from the vehicle without moving (e.g., unloading and reloading) items of other customers to serve later.

The first paper considering multi-drop situations was Bischoff et al. (1995), after which several works followed proposing a variety of techniques to tackle unloading constraints. These papers deal both with the MDCLP and the capacitated VRP with D-dimensional loading constraints (DL-CVRP), where D typically equals 2 or 3. We discuss papers belonging to both streams below.

For the MDCLP, Pan, Chu, Han, & Huang (2009) propose a wall-building loading heuristic while Christensen & Rousœ (2009) design a tree-search based method that uses greedy insertion rules and a dynamic breadth strategy. Liu, Yue, Dong, Maple, & Keech (2011) introduce the concept of *untakeout field*, that is a space in the container where violations of unloading constraints occur, and present a heuristic approach operating on regions to locate items called subvolumes. Junqueira, Morabito, & Yamashita (2012b) and de Queiroz & Miyazawa (2013) develop MILP models for special cases of the MDCLP. Martínez, Alvarez-Valdes, & Parreño (2015) and Iori et al. (2020) solve this problem using GRASP algorithms. In a recent work by Nascimento et al. (2021), twelve practical constraints for the CLP are formulated mathematically, including multi-drop situations and manual unloading constraints, which are treated as hard constraints.

The first papers dealing the with DL-CVRP were Gendreau, Iori, Laporte, & Martello (2006) and Iori, Salazar-González, & Vigo (2007). The former paper proposes a nested tabu search algorithm, while the latter uses branch-and-cut and branch-and-bound to minimize routing costs and check loading feasibility, respectively. For the same problem, Zachariadis, Tarantilis, & Kiranoudis (2009) develop a guided tabu search procedure, while Fuellerer, Doerner, Hartl, & Iori (2009, 2010) employ ant colony optimization. Tabu search, guided tabu search, and ant colony optimization metaheuristics for the DL-CVRP are compared in Iori & Martello (2010). Bortfeldt (2012) develops a hybrid algorithm that uses tabu search for routing and tree search for loading. Exact approaches based on branch-and-cut and MILP models are later presented in Hokama, Miyazawa, & Xavier (2016) and Pollaris, Braekers, Caris, Janssens, & Limbourg (2016), respectively. A branch-and-cut algorithm is also recently used in Ferreira, de Queiroz, & Toledo (2021) to solve a green variant of the DL-CVRP that aims at reducing CO_2 emissions.

The unloading constraints are also referred to as *sequential constraints*, *rear constraints*, or *last-in-first-out* (LIFO) constraints in the literature, but we simply use unloading constraints hereafter. Regardless of the name, the specification of such constraints is not unique but varies depending on the considered application. In particular, we identified four different specifications: (i) *above* constraints, (ii) *visibility* constraints, (iii) *reachability* constraints, and (iv) *separation* constraints. The above constraints are violated if an item of a later customer lies above (even partly) of an item of an earlier customer, requiring the former item to be lifted and relocated. The visibility constraints demand that an item is visible from the container doors when it has to be unloaded, which is a condition needed in practice to unload a box using a forklift. Therefore, no item of a later customer can lie (even partly) between the item to be unloaded and the container doors. The reachability constraints model the ability of a worker to reach an item given a partial packing solution, which may not be possible for a box that is visible. Lastly, the separation constraints force the placement of items belonging to different customers into dedicated regions in the container that are separated by virtual walls.

Most papers that include unloading constraints account for the above and visibility constraints, although not always using these names (Christensen & Rousœ, 2009; Ferreira et al., 2021; Fuellerer, Doerner, Hartl, & Iori, 2010; Gendreau et al., 2006; Hokama et al., 2016; Iori et al., 2020; Iori & Martello, 2010; Iori et al., 2007; Nascimento et al., 2021; Pan et al., 2009; Pollaris et al., 2016; de Queiroz & Miyazawa, 2013), whereas it is less common to account for reachability constraints (Junqueira, Morabito, & Sato Yamashita, 2012a; Liu et al., 2011; Martínez et al., 2015) and separation constraints (Junqueira et al., 2012b; Martínez et al., 2015). We will consider above, visibility, and reachability as soft unloading constraints, and neglect separation constraints as these can only be modeled as hard constraints and are hence not relevant to the scope of our paper.

To the best of our knowledge, unloading constraints have always been treated as hard constraints in the literature (with few exceptions discussed below), which means that no relocation of boxes is allowed during delivery at all. This approach may be too inflexible and result in vehicles carrying significant lower value. Depending on the application, relocating items may be acceptable in practice and thus preferable to reducing cargo value. This entails a trade-off between the transported value and the indirect costs incurred due to relocating items during delivery, which we investigate in this paper. Modeling soft unloading constraints thus adds to the extant literature by providing a more general and flexible definition of the MDCLP. Our penalty functions can be adjusted by the operator depending on the needs, and hard constraints can be used too by setting these penalties to infinity.

Liu et al. (2011) introduced an unloading cost for an item that is proportional to the number of items to relocate in order to unload it, i.e., the items located in their untakeout field, which can be seen as a special case of the penalty functions we use. However, the authors set all such costs equal to zero in their algorithm and experiments, which in effect means enforcing again hard unloading constraints. Lurkin & Schyns (2015) deal with loading a cargo aircraft to serve multiple airports by minimizing the number of handling operations. In this case, the aircraft has fixed slots arranged into rows where to assign the so-called unit loading devices. Thus, although this problem includes soft unloading constraints, it does not embed the same 3D combinatorial complexity as our MDCLP. Moreover, all handling operations count the same. To the best of our knowledge, only Gajda et al. (2022) have not treated the unloading constraints in an MDCLP as hard constraints, but instead allowed for items to be relocated. Nonetheless, any relocation is considered equivalent also in this paper regardless of the volume or weight of the item to move, and the number of moves in a packing solution is only counted a posteriori, i.e., the solution approach does not explicitly minimize it. Therefore, our use of penalty functions in the objective and the explicit consideration of these penalties in our heuristic represents a substantial generalization in terms of both modeling and methodology.

Worth mentioning are finally some works that analyze trade-offs involving unloading constraints. Männel & Bortfeldt (2016) and a follow-up work by the same authors (Männel & Bortfeldt, 2018) provide an interesting perspective on box reloading effort in multi-drop situations. They show that for the pickup and delivery problem, in contrast to the CVRP, a standard LIFO policy is not sufficient to rule out any reloading effort. More constraints are needed to do so, which gives rise to a spectrum of problem variants with different behavior in terms of reloading effort. Junqueira et al. (2012a) study reachability constraints when varying the parameter representing the distance that can be reached, showing that as these constraints become stricter, the packing often becomes less efficient. While the aforementioned variants of unloading constraints affect the feasibility of the solutions, violations of unloading constraints in our work affect the objective function by means of penalties.

2.3. Summary of contributions

The main contributions of this work are the following:

1. We formally study the MDCLP with soft unloading constraints and provide a mathematical programming formulation that accounts for both above and visibility soft constraints.
2. We propose a heuristic framework that includes construction and improvement phases based on a randomized extreme-point heuristic and the reconstruction of packing regions with high penalties, respectively. The heuristic is able to solve large instances and accounts for all types of penalty functions and soft unloading constraints (above, visibility, and reachability).
3. We execute an extensive computational study involving different instances, penalty functions, and region reconstruction methods. Our findings underscore the importance of accounting for soft unloading constraints in the MDCLP and can be relevant for logistics operators.

3. MDCLP with soft unloading constraints

We formally introduce the MDCLP with soft unloading constraints along with the used notation in Section 3.1, derive a mathematical programming formulation of this problem in Section 3.2, and provide an example comparing MDCLP variants with hard, soft, and no unloading constraints in Section 3.3.

3.1. Problem description

We are given a set $\mathcal{B}$ of items/boxes available for loading that are rectangular-shaped, i.e., cuboids. Each item $i \in \mathcal{B}$ is associated with three dimensions: length, width, and height, denoted $(l_i, w_i, h_i) \in \mathbb{R}^3_+$, and hence a volume $v_i = l_i \cdot w_i \cdot h_i$. Moreover, item i has a weight $q_i > 0$, a value $\pi_i > 0$, and belongs to a customer $c_i \in \{1, \dots, M\}$, where 1 is the first customer to serve and M is the last. A single rectangular-shaped container is given with dimensions $(L, W, H) \in \mathbb{R}^3_+$. Boxes can only be loaded in the container orthogonally relative to the boundaries of the container, which is a standard assumption. Each box is also associated with a set of feasible orientations in which it can be rotated and that is a subset of the six possible orthogonal orientations of a cuboid.

We introduce a three-dimensional Cartesian coordinate system (x, y, z) with the origin $(0, 0, 0)$ coinciding with the bottom-left-rear corner of the container. The x-axis is directed along the long side of the container, the y-axis is perpendicular to it on the container's floor, and the z-axis completes the coordinate system in the vertical direction. The four corner points of the container's door where unloading takes place have coordinates $(L, 0, 0)$, $(L, W, 0)$, (L, W, H) and $(L, 0, H)$. Given such a coordinate system, the loading position of an item i inside the container is uniquely defined by the coordinates of its bottom-left-rear corner, denoted by (x_i, y_i, z_i), coupled with a feasible orientation that has the effect of altering the order of the elements in the size vector (l_i, w_i, h_i). Although both our mathematical programming formulation and heuristic account for rotations, to ease exposition we assume in the following that the orientation is fixed, i.e., rotations are not allowed.

As discussed in Section 2.2, we consider above, visibility, and reachability soft unloading constraints. To model them we introduce an overlapping condition for two items $i, j \in \mathcal{B}$ on the x-axis

$$\textsc{Overlap}(i, j, x) \iff \big(x_i \le x_j < x_i + l_i\big) \vee \big(x_j \le x_i < x_j + l_j\big),$$

and use analogous conditions to define overlapping in the other two coordinates. Violations of the unloading constraints for a pair of packed boxes $i, j \in \mathcal{B}$ arise under the following situations.

- *Above violation*: item i has to be unloaded before item j but the latter item is placed above the former, even partially. Mathematically, this means that boxes i and j overlap in both x and y coordinates and j is placed at a higher z coordinate than i (Nascimento et al., 2021), i.e.,

$$\big(c_i < c_j\big) \wedge \big(z_j \ge z_i + h_i\big) \wedge \textsc{Overlap}(i, j, x) \wedge \textsc{Overlap}(i, j, y). \tag{1}$$

  When condition (1) is true, we assign a cost/penalty $f_A(q_j, v_j, z_j) \ge 0$ to the above violation that is a function of weight, volume, and z coordinate of the item j to relocate. Intuitively, this penalty should be non-decreasing in all three features since the heavier, the bulkier, or the higher the item was loaded, the harder or time consuming will be to reposition it.
- *Visibility violation*: item i has to be unloaded before item j but the latter is placed, even partially, between the unloading side and item i thereby making i not (entirely) visible from the entrance of the container. Mathematically, this happens when boxes i and j overlap in y and z directions and j is placed at a larger x coordinate than i, i.e.,

$$(c_i < c_j) \wedge \big(x_j \ge x_i + l_i\big) \wedge \textsc{Overlap}(i, j, y) \wedge \textsc{Overlap}(i, j, z). \tag{2}$$

  When condition (2) holds, i.e., visibility constraints are violated for item i, we assign a penalty $f_V(q_j, v_j, z_j) \ge 0$ with analogous structure to the penalty related to the above constraints.
- *Reachability violation*: item i has to be unloaded but is unreachable by a human operator (or a machine/forklift) since the distance between this item and the position the operator can reach is higher than a fixed quantity (Junqueira et al., 2012a; Liu et al., 2011). Specifically, assume j is the first box that prevents the operator from moving further ahead and reaching box i. Then, i is unreachable if

$$(c_i < c_j) \wedge \big(\delta_{ij} \ge \min\{H_{touchable} - z_i, L_{touchable}\}\big), \tag{3}$$

  where $\delta_{ij} := x_j + l_j - (x_i + l_i)$ is the distance between boxes i and j seen from the unloading side, and $L_{touchable}$ and $H_{touchable}$ describe respectively the length of the worker's arm and the height the worker's hands can reach. Condition (3) characterizes (un)reachability as a function of both the distance δ_{ij} and the z coordinate of the item to pick. It holds when δ_{ij} is larger than $L_{touchable}$, or when this distance plus z_i is larger than $H_{touchable}$. $L_{touchable}$ and $H_{touchable}$ are typically set to 60 and 200 cm, respectively. Under this choice,

the min function evaluates to 60 cm for $z_i \in [0, 140]$ cm and decreases linearly for values of $z_i > 140$ cm, reaching zero for $z_i = 200$ cm. See also Martínez et al. (2015) for some illustrations of these situations. When reachability is violated, we assign a penalty function $f_R(q_j, v_j, z_j, \delta_{ij})$, which may depend on the distance δ_{ij} in addition to the properties of the box j to relocate.

The MDCLP with soft unloading constraints is defined as the problem to select and orthogonally pack items from $\mathcal{B}$ inside the container such that the total value of the loaded items discounted by penalties due to violations of conditions (1)–(3) is maximized. Notice that we do not include in our problem statement additional practical constraints, since we want to isolate the effect of the unloading constraints, and highlight the advantages of modeling them as soft by means of penalty functions. Therefore, our solutions do not ensure that items are, e.g., well supported and vertically stable during the delivery process, or that weight is balanced in the container. Practical implementations of our approach should of course include the constraints needed for the specific application.

3.2. Mathematical programming model

We provide next a mathematical programming formulation of the MDCLP with soft unloading constraints covering the case of above and visibility constraints. Although reachability constraints are considered in our heuristic framework in Section 4, we found it challenging to characterize the first item j blocking the path of the operator to reach another item i in a mathematical program, hence we neglect these constraints here. As we did in Section 3.1, to simplify exposition we omit item orientations when describing the model and relegate a full formulation including rotations to an appendix.

For each box $i \in \mathcal{B}$, in addition to the continuous position variables (x_i, y_i, z_i) we define a binary variable t_i equal to 1 if item i is loaded and 0 otherwise. Moreover, to ensure that item pairs do not overlap, we introduce for $i, j \in \mathcal{B}$ binary variables b_{ij}, f_{ij}, and u_{ij} equal to 1 if item i is fully behind, left, or under item j, respectively, and 0 otherwise. At least one of these variables must be 1 when both i and j are loaded (i.e., $t_i = t_j = 1$), which translates to the non-overlapping constraints (4).

$$b_{ij} + b_{ji} + f_{ij} + f_{ji} + u_{ij} + u_{ji} + (1 - t_i) + (1 - t_j) \geq 1 \quad \forall i, j \in \mathcal{B},\ i < j, \tag{4a}$$

$$x_i + l_i \leq x_j + L(1 - b_{ij}) \quad \forall i, j \in \mathcal{B}, \tag{4b}$$

$$y_i + w_i \leq y_j + W(1 - f_{ij}) \quad \forall i, j \in \mathcal{B}, \tag{4c}$$

$$z_i + h_i \leq z_j + H(1 - u_{ij}) \quad \forall i, j \in \mathcal{B}. \tag{4d}$$

We further restrict the domain of the position variables $x_i \in [0, L - l_i]$, $y_i \in [0, W - w_i]$, and $z_i \in [0, H - h_i]$ to guarantee that loaded items do not exceed the boundaries of the container.

Constraints (4) are common in packing problems (Chen, Lee, & Shen, 1995). Due to (4b), $b_{ij} = 1 \implies x_i + l_i \leq x_j$, hence the two items do not overlap as i is fully behind j on the x coordinate (similarly for the other coordinates). These constraints alone however do not allow tracking the situations when two items do overlap, which is needed to model the unloading constraints. To this end, we introduce a second set of constraints (5) that enforce the reverse implications, e.g., $b_{ij} = 0 \implies x_j \leq x_i + l_i$.

$$x_j \leq x_i + l_i + L\, b_{ij} \quad \forall i, j \in \mathcal{B},\ c_i \neq c_j, \tag{5a}$$

$$y_j \leq y_i + w_i + W\, f_{ij} \quad \forall i, j \in \mathcal{B},\ c_i \neq c_j, \tag{5b}$$

$$z_j \leq z_i + h_i + H\, u_{ij} \quad \forall i, j \in \mathcal{B},\ c_i \neq c_j. \tag{5c}$$

Constraints (4), (5) jointly model $b_{ij} + b_{ji} = 0 \iff \textsc{Overlap}(i, j, x)$, and similarly $f_{ij} + f_{ji} = 0 \iff \textsc{Overlap}(i, j, y)$, and $u_{ij} + u_{ji} = 0 \iff \textsc{Overlap}(i, j, z)$. Notice that generating constraints (5) is not necessary for item pairs belonging to the same customer (i.e., $c_i = c_j$).

To track joint overlap in two dimensions in above and visibility constraints (1), (2), we introduce support binary variables $a_{ij}, d_{ij} \in \{0, 1\}$ so that $a_{ij} = 1 \iff \textsc{Overlap}(i, j, x) \wedge \textsc{Overlap}(i, j, y)$ and $d_{ij} = 1 \iff \textsc{Overlap}(i, j, y) \wedge \textsc{Overlap}(i, j, z)$. This can be done by imposing

$$b_{ij} + b_{ji} + f_{ij} + f_{ji} \geq 1 - a_{ij} \quad \forall i, j \in \mathcal{B},\ c_i < c_j, \tag{6a}$$

$$b_{ij} + b_{ji} + f_{ij} + f_{ji} \leq 2(1 - a_{ij}) \quad \forall i, j \in \mathcal{B},\ c_i < c_j, \tag{6b}$$

$$f_{ij} + f_{ji} + u_{ij} + u_{ji} \geq 1 - d_{ij} \quad \forall i, j \in \mathcal{B},\ c_i < c_j, \tag{6c}$$

$$f_{ij} + f_{ji} + u_{ij} + u_{ji} \leq 2(1 - d_{ij}) \quad \forall i, j \in \mathcal{B},\ c_i < c_j. \tag{6d}$$

Moreover, detecting above and visibility violations requires coupling the described overlapping conditions in two directions with another condition for the third coordinate, which is done by

$$p_{ij} + 1 \geq a_{ij} + u_{ij} \quad \forall i, j \in \mathcal{B},\ c_i < c_j, \tag{7a}$$

$$r_{ij} + 1 \geq d_{ij} + b_{ij} \quad \forall i, j \in \mathcal{B},\ c_i < c_j, \tag{7b}$$

where p_{ij} and r_{ij} are additional binary variables that are activated (i.e., set to 1) when above and visibility constraints are violated, respectively. Thus, the MDCLP with soft unloading constraints obeys (4)–(7) and has an objective function

$$\max \sum_{i \in \mathcal{B}} \pi_i t_i - \sum_{i \in \mathcal{B}} \sum_{j \in \mathcal{B}: c_i < c_j} \left[p_{ij} \cdot f_A(q_j, v_j, z_j) + r_{ij} \cdot f_V(q_j, v_j, z_j) \right]. \tag{8}$$

In general, (4)–(8) is a mixed-integer nonlinear program, where nonlinearities only appear in the objective function. If both f_A and f_V are linear functions of box parameters like weight q_j and volume v_j, and of the loading coordinate z_j, then this model becomes an MILP that can be solved using off-the-shelf commercial optimization solvers. Examples of such linear penalties are

$$f_A(q_j, v_j, z_j) := \alpha_A q_j + \beta_A v_j + \gamma_A z_j + \eta_A, \tag{9a}$$

$$f_V(q_j, v_j, z_j) := \alpha_V q_j + \beta_V v_j + \gamma_V z_j + \eta_V, \tag{9b}$$

with $(\alpha_A, \beta_A, \gamma_A, \eta_A, \alpha_V, \beta_V, \gamma_V, \eta_V) \in \mathbb{R}^8_+$. We report the MILP including rotations in an appendix.

Notice that a formulation of the MDCLP with hard above/visibility unloading constraints can be obtained as a special case of (4)–(8) by fixing all penalty activation variables p_{ij} and r_{ij} to zero. This implies that $a_{ij} + u_{ij} \leq 1$ and $d_{ij} + b_{ij} \leq 1$ in (7), i.e., violations are infeasible.

Table 1
Parameters defining the MDCLP instance.

Demand	Length (l_i, cm)	Width (w_i, cm)	Height (h_i, cm)	Volume (v_i, m^3)	Weight (q_i, ton)	Value (π_i)	Customer (c_i)
2	95	50	35	0.166	0.22	0.22	1
2	90	55	45	0.223	0.24	0.24	2
2	90	60	40	0.216	0.26	0.26	3
2	105	65	40	0.273	0.28	0.28	4

Table 2
Results from the MILP formulations.

		Cargo		Penalty		
Strategy	Objective	Value	# items	Amount	# violations	Runtime (s)
MDCLP−B	74.2	**88.0**	7	13.8	5	0.8
MDCLP−H	76.0	76.0	6	**0.0**	0	8.6
MDCLP−S	**80.9**	86.0	7	5.1	2	22.1

3.3. Illustrative example

In this section, we use the model formulation introduced in Section 3.2 to compare three strategies to tackle the MDCLP: (i) solve the basic CLP without unloading constraints and assess the penalties a posteriori (denoted MDCLP−B strategy), (ii) solve the MDCLP with hard unloading constraints (MDCLP−H), and (iii) solve directly the MDCLP with soft unloading constraints (MDCLP−S). Notice that also the first two strategies provide a feasible solution to the MDCLP with soft unloading constraints.

To illustrate this comparison, we define in Table 1 a small instance with 8 items and 4 customers, where up to 2 identical items need to be delivered to each customer. The container size is $(L, W, H) = (160, 120, 100)$ cm and all six orientations of the items are permissible.

We consider model (4)–(8) or simplifications thereof using penalties (9) for above and visibility constraints with coefficients $(\alpha_A, \beta_A, \gamma_A, \eta_A, \alpha_V, \beta_V, \gamma_V, \eta_V) = (0.1, 0.1, 0, 0, 0.1, 0.1, 0, 0)$, i.e., penalties are linear functions of weight and volume of the boxes to move and include no dependency on the z coordinate ($\eta_A = \eta_V = 0$) and no constants ($\gamma_A = \gamma_V = 0$). Penalties in this example are defined in a simplistic manner as the focus here is on comparing the three strategies. In practice, however, defining the functional form of the penalties and their coefficients plays a bigger roles because, depending on the unloading system (e.g., human worker or forklift) or even the specific product, weight can be a strong constraint, or volume can be, or both. We solve the corresponding MILPs to optimality using Gurobi 9.1 and show the results in Table 2 and Fig. 1. To ease intuition, columns in this table related to the objective function value and its two components (cargo value and penalty amount) are normalized to 100% of the total value of the 8 items, that is, a solution including all items and with zero penalty would have an objective of 100.

Interestingly, the three packing solutions are different under each approach in terms of both cargo composition and loading coordinates of the chosen boxes. While MDCLP−B achieves the highest cargo value of 88.0 (which is expected as this strategy indeed maximizes this objective alone), it violates 5 times the above and visibility constraints incurring a high penalty of 13.8. The opposite can be said for MDCLP−H, which does not violate any of the unloading constraints but results in the lowest cargo value of 76.0, with only 6 boxes loaded instead of 7 as under the other strategies. Finally, MDCLP−S achieves a cargo value of 86.0 and violates 2 unloading constraints for a total penalty of 5.1, hence reaching the best objective function value of 80.9. Therefore, by accounting for soft unloading constraints, MDCLP−S manages the trade-off between cargo value and penalties more efficiently and its objective value is 9.0% and 6.4% higher than that of MDCLP−B and MDCLP−H, respectively.

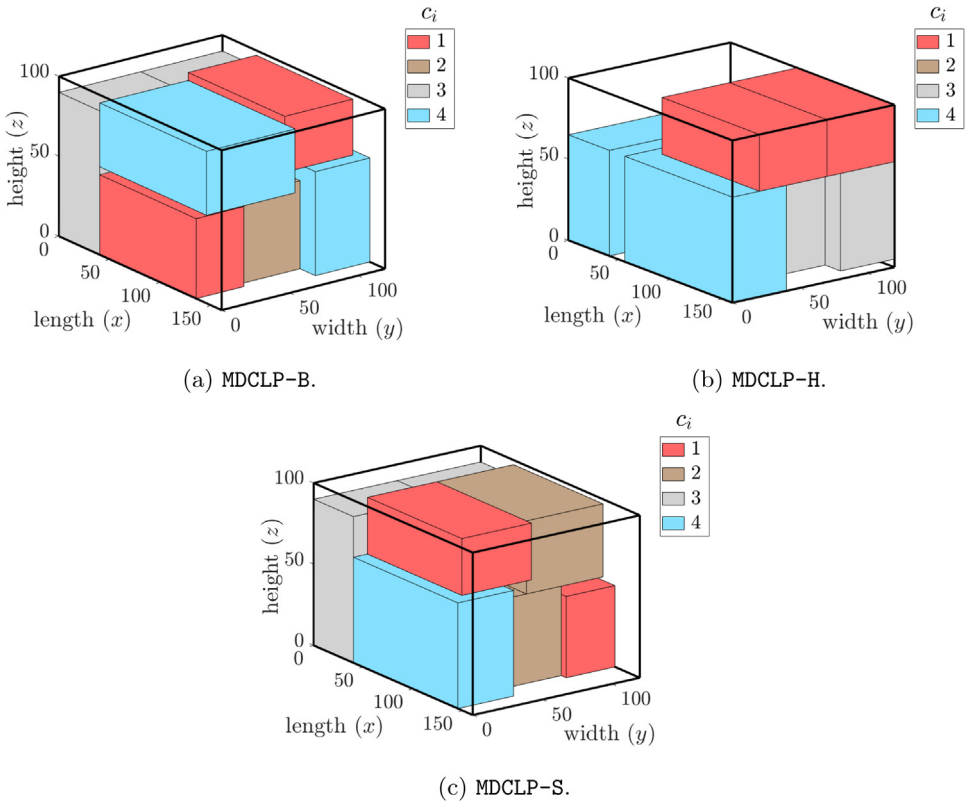
The running time varies considerably across formulations. Whilst MDCLP−B needs less than one second, MDCLP−H and MDCLP−S take roughly 9 and 22 seconds respectively, which is substantial given that this small instance only includes eight items. The reason behind this variation is that modeling soft unloading constraints requires a significant number of additional binary variables and big-M constraints compared to the standard CLP (which is already challenging as discussed in Section 2), namely variables a_{ij}, d_{ij}, p_{ij}, and r_{ij} appearing in constraints (5)–(7). The formulation with hard unloading constraints lies somehow in between, since a_{ij} and d_{ij} are needed, but not p_{ij} and r_{ij}. Additional experiments show that solving times are highly instance-dependent and vary depending on both the dimensions of the boxes and the coefficients of the penalties (as also discussed later in Section 5.4). Nonetheless, the observed running times clearly indicate a consistent ranking of the three strategies MDCLP−B, MDCLP−H, and MDCLP−S as mentioned above, and that it is not viable to solve the MDCLP with hard or soft unloading constraints optimally for larger instances, e.g., comprised of 50 or more items. Moreover, the presented formulation does not include reachability constraints and would be even harder to tackle under nonlinear penalties. For these reasons, we develop next a heuristic framework that sacrifices optimality but is very fast and can deal with large instances as well as more general penalties. Despite our focus is on the unloading constraints, this heuristic could be extended to incorporate other practical constraints (e.g., vertical stability and load bearing).

4. Heuristic algorithm

At a high level, our heuristic algorithm applies in sequence construction and improvement methods that iteratively perform randomized loading operations to build or improve a solution. Construction and improvement phases of our heuristic are described in Sections 4.1 and 4.2, respectively.

4.1. Construction phase

Our construction algorithm is based on the concept of the extreme points (EPs) from Crainic et al. (2008). The idea underlying the approach is simple and consists in inserting boxes sequentially in the container one after the other, starting from the empty container, and using the EPs as the set of candidate locations for new insertions. Since constructing a solution in this manner is quick, we can iterate the procedure N^C times to generate different packing solutions, then pick the one with the highest objective value. Constructing one packing solution requires defining: (i) how to sort



(a) MDCLP-B. (b) MDCLP-H.

(c) MDCLP-S.

Fig. 1. Illustration of the three loading solutions (see web version for interpretation of colors).

the items, (ii) how to randomize the sorted list, (iii) how to select the best point from a set of EPs, and (iv) how to update the set of EPs after each insertion. We explain our choices in this regard next.

Since the solution of a constructive method depends on the order in which items are considered, we initially sort them according to a criterion that facilitates the subsequent loading, e.g., by placing first those items that are larger and hence harder to pack later when space is limited. Specifically, we sort by non-increasing values one of the following six attributes: volume (v_i), height (h_i), area ($l_i \cdot w_i$), customer-volume (c_i, then v_i), customer-height (c_i, then h_i), and customer-area (c_i, then $l_i \cdot w_i$). In the latter three cases, items are sorted first according to the primary attribute c_i; items with equal c_i are then sorted by a secondary attribute. We randomly pick one out of these six sorting criteria for each solution we construct, i.e., each iteration $n = 1, \ldots, N^{\mathrm{C}}$. While the first three criteria based on box size are standard (Crainic et al., 2008), sorting by customer number in the other three cases favors the placement of boxes with higher and lower customer numbers deeper in the container and closer to its doors, respectively. Thus, these criteria implicitly favor generating solutions with fewer violations of the unloading constraints (Gajda et al., 2022).

Once the items are sorted, we randomize the sequence to introduce diversification during loading and produce each iteration a different solution. We do this by swapping the position of items that are "similar" according to the attribute θ used for sorting. Specifically, following Trivella & Pisinger (2016), we define the probability of swapping two consecutive items i and $i+1$ in the sequence by

$$\Pr(i \leftrightarrow i+1) := \begin{cases} 5\,(\theta_{i+1}/\theta_i - 0.9), & \text{if } \theta_{i+1}/\theta_i > 0.9, \\ 0, & \text{otherwise.} \end{cases}$$

This formula states that two items with identical sorting attribute, i.e. such that $\theta_i = \theta_{i+1}$, are swapped position with 50% probability. As the ratio θ_{i+1}/θ_i decreases (i.e., the items become more different), the probability $\Pr(i \leftrightarrow i+1)$ decreases too, until it reaches 0% for $\theta_{i+1}/\theta_i = 0.9$. In case of combined attributes, θ_i equals the secondary attribute. Furthermore, we randomly rotate the items by picking one of their feasible orientations with equal probability. This randomization is not carried out separately for each item but in a block-wise fashion, that is, the same rotation is applied to all items with identical dimensions, thereby easing the formation of compact blocks with no holes and hence better utilizing the space in the container (Gajda et al., 2022).

Once sorting and randomization have been executed, the items are loaded into the container one by one. For each insertion, one EP from the available set has to be selected and the box placed so that its bottom-left-rear corner coincides with that EP. Initially, the list of EPs is made of just the bottom-left-rear corner of the container with coordinates (0,0,0). When a box is loaded, the set of available EPs is updated by removing the chosen point while generating up to six new EPs using the algorithm described in Crainic et al. (2008), which exploits projections of the corners of the newly inserted box onto the walls of the container or other items already loaded. To select the EP where to place an item, we adopt a first-fit decreasing approach that considers the EP with the lowest x-coordinate, then the lowest y-coordinate in case of ties on x, and finally the lowest z-coordinate in case of ties on both x and y. Using such a first-fit decreasing strategy, we pick the first EP which is feasible for placing the current box, that is, the box must be fully contained in the container and must not overlap with the boxes already loaded up to that point.

If an item i cannot be placed in any of the EPs available when i is considered, (i.e., all EPs are infeasible for inserting i), then this item is temporarily stored in a *retry list*. After attempting to load all items in the container, a second round is performed with the retry list. Thus, the goal of such a list is to reattempt loading when new EPs have been generated. Since the number of items in the

retry list is typically much smaller than $|\mathcal{B}|$, to increase the chances of loading we adopt an enhanced first-fit decreasing variant that checks feasibility not only for the predefined (randomized) orientation, but for all feasible orientations. Boxes that fail insertion again are discarded from the constructive solution but may be reconsidered during the improvement phase presented next.

4.2. Improvement phase

The unloading constraints are not taken into account explicitly during construction, but only implicitly by including the customer c_i as a sorting criterion. Our improvement phase is instead designed to explicitly reduce the penalties due to violations of the unloading constraints. Indeed the algorithm iteratively defines a region in the container with high improvement potential, empties it, and refills it with the goal of increasing the objective function value. Region reconstruction is one of the five possible improvement moves in the VNS presented by Parreño et al. (2010). In our context, specifying promising regions is guided by indicators that quantify the amount of penalties in the region and/or the proportion of empty spaces. Thus, rebuilding these regions has the potential to reduce penalties and/or pack more items in the available space. The reconstructed solution is then compressed to a gapless packing and the gained space is used to load additional items. The structure of our approach is outlined in Algorithm 1. Given a packing solution $\mathcal{S}$, this algorithm attempts to improve it by reconstructing regions iteratively for up to N^{I} iterations (alternatively, until a given running time is reached)[1]. In the following we explain in detail the steps underlying each iteration.

Algorithm 1: Improvement algorithm.

Input: Packing solution $\mathcal{S}$; maximum number of iterations N^{I}

for iteration $n = 1$ to N^{I} **do**

> **Step 1.** Compute matrices of conflicts (C), penalty density (D) and empty spaces (E)
>
> **Step 2.** Select item pair $i, j \in \mathcal{S}$ delimiting the region using criterion `M-1`, `M-2`, or `M-3`
>
> **Step 3.** Define $\mathcal{S}^0$ obtained by removing all items intersecting cuboid $\mathfrak{P}(i, j)$ from $\mathcal{S}$
>
> **Step 4.** Refill $\mathcal{S}^0$ using best-fit decreasing and items in $\mathcal{B} \setminus \mathcal{S}^0$, obtaining $\mathcal{S}'$
>
> **Step 5.** Compress $\mathcal{S}'$ and refill it with best-fit decreasing and items in $\mathcal{B} \setminus \mathcal{S}'$, getting $\mathcal{S}''$
>
> **Step 6.** Set $\mathcal{S}$ as the solution with highest objective value among $\{\mathcal{S}, \mathcal{S}^0, \mathcal{S}', \mathcal{S}''\}$

Output: Improved packing solution $\mathcal{S}$

Step 1 computes three matrices to support identifying a promising region: the *conflict matrix* C, the *density matrix* D and the *empty space matrix* E, each with size $|\mathcal{S}| \times |\mathcal{S}|$. Matrix C evaluates conflicts arising between item pairs. Specifically, given $i, j \in \mathcal{S}$ with $c_i < c_j$, we define $C_{ij} := f_{\mathrm{V}}(q_j, v_j, z_j) + f_{\mathrm{A}}(q_j, v_j, z_j) + f_{\mathrm{R}}(q_j, v_j, z_j, \delta_{ij})$, which equals zero if above, visibility, and reachability constraints are respected between i and j, and the incurred penalty otherwise. Matrix D is more sophisticated and encodes information about the density of penalties inside a given volume. For each $i, j \in \mathcal{S}$, we compute the smallest orthogonal cuboid that entirely includes both boxes, which we denote by $\mathfrak{P}(i, j)$. Then, for each item $k \in \mathcal{S}$ which overlaps with $\mathfrak{P}(i, j)$, we calculate the unloading cost generated by this item by summing up the k-th column of matrix C, and multiply it by the proportion of box k's volume that falls inside $\mathfrak{P}(i, j)$. We define D_{ij} as the sum over k of these penalties divided by the volume of $\mathfrak{P}(i, j)$. Finally, matrix E describes the amount of empty space in regions of the container. For $i, j \in \mathcal{S}$, E_{ij} is defined as the proportion of volume of $\mathfrak{P}(i, j)$ that is not occupied by boxes. Note that matrices D and E are symmetric while C is not.

Step 2 uses these matrices to select the region to reconstruct, which involves choosing two items $i, j \in \mathcal{S}$ and computing the associated cuboid $\mathfrak{P}(i, j)$. We define three approaches to select i and j. The first approach, denoted `M-1`, just uses the conflict matrix C. It computes unloading costs originating from each item by summing up the columns of C, and randomly picks one of the ξ items with highest unloading cost. For this item j, a second item i is chosen randomly among those in conflict with j, i.e., such that $C_{ij} > 0$. The second approach `M-2` uses the density matrix D by randomly selecting one of the ξ item pairs (i, j) with highest D_{ij} value. Finally, the last approach `M-3` generalizes the former one by exploiting both penalty and empty space information. This approach randomly picks one of the ξ item pairs (i, j) with highest $\zeta^{\mathrm{D}} D_{ij} + \zeta^{\mathrm{E}} E_{ij}$ value, where ζ^{D} and ζ^{E} are trade-off parameters. In all three methods, ξ governs the level of randomization and the volume of $\mathfrak{P}(i, j)$ is constrained not to exceed half of the container's volume.

Step 3 removes from $\mathcal{S}$ all items that even partially overlap with $\mathfrak{P}(i, j)$, obtaining a new solution $\mathcal{S}^0$. Although the latter carries lower cargo value, it may also have fewer penalties.

Step 4 refills the emptied region by applying an extreme-point heuristic. First, some EPs are (re)generated at the bottom-left-rear corners of the removed items. Then, all EPs are ranked in a first-fit fashion based on coordinates as described in Section 4.1 and are considered sequentially for placing items. For each insertion, we evaluate all items $i \in \mathcal{B} \setminus \mathcal{S}^0$ and all feasible orientations $o \in \mathcal{R}_i$ using a best-fit strategy with a merit function $m_{io} := \pi_i - \left\{ \sum (C_{i^o j} + C_{j i^o}) : j \in \mathcal{S}^0 \right\}$, where $C_{i^o j}$ denotes the penalty between item i with orientation o and the already loaded item j, i.e., it coincides with the conflict matrix element C_{ij} once item i is given orientation o (analogous for $C_{j i^o}$). The item-orientation pair (i, o) with highest merit value is selected for loading. The procedure ends when all items are loaded or when all EPs are considered, giving a new solution $\mathcal{S}'$.

Step 5 transforms $\mathcal{S}'$ into a so-called *gapless* packing where no box can be moved to lower x, y, or z coordinates, i.e., all boxes are shifted as much as possible towards the corner (0,0,0). This is important to reduce empty spaces since the refilling process often leads to gaps between items. To produce such a gapless packing, we use the *interval graphs* resulting from the projection of boxes onto the Cartesian axes (Fekete et al., 2007), which allows to easily identify the sequence in which boxes must be shifted (Trivella & Pisinger, 2016). To illustrate, consider the interval graph related to coordinate x. Each node in this graph is a box and an edge connects two nodes if the corresponding boxes overlap in x. Each edge is also assigned a direction, with the arrow pointing towards the box with highest x coordinates. We can use this graph to perform a shift by reassigning coordinates using two rules. First, a box i without incoming edges is given a coordinate of $x_i = 0$ and is marked as processed. Second, a box with only incoming edges from processed boxes is assigned coordinate x_i equal to the maximum of $x_j + l_j$ over the connected processed boxes $j \in \mathcal{S}'$. Using these rules, we can iteratively process all nodes. The same procedure is applied to the y and z directions. As the described compression method may free up some space in the container, we

[1] A packing solution $\mathcal{S}$ is given by the set of loaded items, their placement coordinates, and their orientation. To ease exposition, when it is clear from the context, we abuse notation and refer to $\mathcal{S} \subseteq \mathcal{B}$ as just the loaded item set.

apply again the extreme-point best-fit strategy to attempt filling this space. We call the resulting solution $\mathcal{S}''$.

Step 6 picks the best solution among $\{\mathcal{S}, \mathcal{S}^0, \mathcal{S}', \mathcal{S}''\}$, i.e., the one with highest objective function value. Conceptually, it is not possible to rank these four solutions a priori. For example, we know that the cargo value under $\mathcal{S}''$ is at least as high as the one under $\mathcal{S}'$, but the former solution may incur higher penalties because more items are loaded, hence more easily some of the unloading constraints are violated. Finally, the entire process (Steps 1–6) is iterated by selecting and refilling a new region, which will be different than the previous one due to the embedded randomization.

5. Numerical study

This section presents an extensive computational study based on the heuristic algorithm of Section 4. In Section 5.1, we introduce the instances and computational setup. In Section 5.2, we compare MDCLP variants with soft, hard, and no unloading constraints. In Section 5.3, we study in detail the effectiveness of the improvement phase. In Sections 5.4 and 5.5 we compare the performance of our method with that of a commercial optimization solver and a heuristic algorithm from the literature, respectively. Finally, we analyze in Section 5.6 the region reconstruction methods used in the improvement phase.

5.1. Instances and study design

Our experiments are based on the well-known BR instances from Bischoff et al. (1995) and Davies & Bischoff (1999), comprising 1500 CLP instances divided into 15 classes of 100 instances each. The 15 classes differ by the level of heterogeneity in the item set, which ranges from weakly heterogeneous (BR1 has $T = 3$ item types) to highly heterogeneous (BR15 has $T = 100$ item types), where an *item type* describes a set of items with identical features, i.e., dimensions and set of permissible orientations. These instances include 178.5 items on average with a maximum of 1961.

A BR instance specifies a container size (which is $587 \times 233 \times 220$ cm in all instances corresponding roughly to a 20-foot container) and a set of T item types each provided with a demand, i.e., the number of identical items. The BR instances are traditionally used to maximize the loaded volume and no data on item value and weight is given, which is instead needed here. Thus, we associate each item type with a value and a weight by multiplying its volume by a random number drawn uniformly from [0.7,1.3] and [1,3], respectively. This randomization is used to create more realistic instances, as not every item has equal density and value while it is reasonable to assume that volume is correlated with both value and weight. We then generate customer numbers by randomly picking a number of customers $M \in \{2, \dots, \min\{8, T\}\}$ and associating all items of the same type with a random customer in $\{1, \dots, M\}$, also ensuring that each customer is assigned at least one item type.

We solved all 1500 instances using the three strategies already introduced in Section 3.3, namely `MDCLP-B`, `MDCLP-H`, and `MDCLP-S`, with small adaptations to the heuristic framework. Specifically:

- `MDCLP-B` (*Sequential approach*). This strategy runs the construction phase only of the heuristic for 30 seconds per instance (see Section 4.1). It only accounts for unloading implicitly through sorting. The penalties are evaluated a posteriori for all generated solutions, and the one with highest objective (total cargo value minus penalties) is selected. Under very high penalties this evaluation may be negative; we choose zero in this case, which corresponds to an empty shipment.
- `MDCLP-H` (*Hard unloading constraints*): This approach relies on an adaptation of the construction phase that enforces hard unloading constraints by only allowing placements of items into EPs when no unloading constraint is violated. As penalties are always zero by definition, the constructed solution with the highest cargo value after a 30-second execution is chosen.
- `MDCLP-S` (*Soft unloading constraints*): This strategy employs the same construction phase as `MDCLP-B` followed by an improvement phase (see Section 4.2). For improvement, we apply Algorithm 1 to the five best solutions returned by the construction phase for 20 seconds each, totaling 100 seconds. We then choose the best among the five improved solutions and the `MDCLP-H` solution, which we recall is always feasible. This means that `MDCLP-S` runs in 160 seconds as it requires the construction solutions from the other two strategies in addition to the improvement phase.

Table 3
Specification of penalty function parameters.

Set	Penalty level	α	β	γ
P-1	low	$2.5 \cdot 10^{-3}$	$5.0 \cdot 10^{-3}$	$4.5 \cdot 10^{-3}$
P-2	medium	$7.5 \cdot 10^{-3}$	$15.0 \cdot 10^{-3}$	$4.5 \cdot 10^{-3}$
P-3	high	$12.5 \cdot 10^{-3}$	$25.0 \cdot 10^{-3}$	$4.5 \cdot 10^{-3}$
P-4	very high	$17.5 \cdot 10^{-3}$	$35.0 \cdot 10^{-3}$	$4.5 \cdot 10^{-3}$

The penalties for violating the unloading constraints are defined as follows (see also Section 3.1).

$$f_{\text{A}}(q_j, v_j, z_j) = (1 + \gamma z_j) \cdot (\alpha q_j + \beta v_j), \tag{10a}$$

$$f_{\text{V}}(q_j, v_j, z_j) = (1 + \gamma z_j) \cdot (\alpha q_j + \beta v_j), \tag{10b}$$

$$f_{\text{R}}(q_j, v_j, \delta_{ij}) = (1 + \gamma \delta_{ij}) \cdot (\alpha q_j + \beta v_j), \tag{10c}$$

which depend on the three parameters α, β, and γ. We consider four different sets of parameters ranging from low to very high penalties and that we report in Table 3.

To elaborate, we set the coefficient $\beta = 2\alpha$ because items' weights are generated by multiplying their volume by a uniform random variable $\mathcal{U}_{[1,3]}$, which has a mean of 2. Thus, we try to attribute equal importance to weight and volume when calculating the penalties. When moving from P-1 to P-4, the coefficients α and β increase linearly and by up to seven times, which based on experimentation appear to cover a reasonable spectrum of values. Regarding γ, this parameter is set in all cases equal to the multiplicative inverse of the height H of the container. This implies that an obstructing item placed close to the top of the container will generate roughly double the penalty compared to the same item placed at the bottom. Consider for example a typical item of value 100, volume 100, and weight 200, neglecting units. Based on Table 3, a single violation of the above constraints (10a) caused by this item would result in a penalty ranging from 0.5–1.0% of its value under P-1, to 3.5–7.0% of its value under P-4, depending on the height at which this box is loaded. Of course, an item may be simultaneously subject to more penalties arising from conflicts with multiple items. In contrast to the illustrative example in Section 3.3, notice that (4)–(8) would be a nonlinear program under penalties (10).

The heuristic algorithm was implemented in Python and run on a server equipped with a quad-core Intel Xeon E3-1585L v5 processor, with memory usage never exceeding 500 MB. On average across all instances, running the construction phase for 30 seconds allows to perform 420 iterations (i.e., packing solutions generated). Due to the large number of randomized iterations we can execute

Table 4
Comparison of MDCLP−H and MDCLP−S strategies.

	MDCLP−H	MDCLP−S				Δ_{SH} (%)			
Class	P−*	P−1	P−2	P−3	P−4	P−1	P−2	P−3	P−4
BR1	77.1	84.1	81.4	80.4	79.9	9.0	5.5	4.2	3.6
BR2	76.2	83.4	80.6	79.4	78.7	9.4	5.8	4.2	3.3
BR3	72.8	80.9	77.0	75.3	74.5	11.2	5.8	3.5	2.4
BR4	72.4	80.4	76.5	75.0	74.4	11.0	5.6	3.6	2.6
BR5	72.1	79.6	75.6	74.0	73.2	10.4	4.8	2.7	1.6
BR6	70.7	79.0	74.6	72.8	72.0	11.7	5.5	3.0	1.8
BR7	70.0	78.1	73.7	72.3	71.1	11.6	5.3	3.3	1.7
BR8	69.4	77.7	73.4	71.4	70.4	12.0	5.9	3.0	1.6
BR9	69.4	77.0	72.7	70.9	70.0	11.0	4.7	2.2	1.0
BR10	68.9	76.5	72.2	70.4	69.4	11.0	4.8	2.1	0.8
BR11	69.3	76.4	72.4	70.4	69.8	10.4	4.5	1.7	0.7
BR12	68.4	76.1	71.4	69.6	68.8	11.2	4.4	1.6	0.6
BR13	68.9	76.3	72.0	70.0	69.2	10.7	4.4	1.6	0.4
BR14	68.3	75.8	71.4	69.4	68.6	11.1	4.5	1.6	0.4
BR15	68.7	75.9	71.4	69.6	68.9	10.5	4.0	1.3	0.3
Mean	70.8	78.5	74.4	72.7	71.9	10.8	5.0	2.6	1.5

in 30 seconds, increasing further the time limit does not bring significant extra benefit to the construction phase. Instead, about 20 region reconstruction iterations can be executed when improving a solution for 20 seconds. An improvement iteration is therefore more costly than a construction iteration, which is expected since Algorithm 1 employs more sophisticated operations than construction. Both construction and improvement iterations could be parallelized to drastically reduce the runtime, which we did not do here. In Sections 5.2–5.5, we use the region definition M−1 with $\xi = 5$ (see Step 2 of Algorithm 1), and compare it with the other variants M−2 and M−3 later in Section 5.6.

5.2. Sequential approach, hard, and soft unloading constraints

We start by comparing the objective function value under MDCLP−H and MDCLP−S for all penalties in Table 4. The reported values are averages across 100 instances of each BR class, except for the last row where the average is over both instances and classes. The MDCLP−H strategy is not affected by the choice of the penalty and its objective is independent from the penalty. The last four columns labeled Δ_{SH} (%) contain the average improvement achieved by MDCLP−S over MDCLP−H as percentage.

Varying the penalty coefficients (α, β) considerably affects the MDCLP−S objective, which decreases with increasing coefficients. This is expected since the set of feasible packing solutions and their cargo value is the same regardless of the soft unloading constraints but the penalty component is an increasing function of α and β. As a consequence, the improvement over MDCLP−H (which is independent of the penalty) varies too and ranges from 9–12% under P−1 to 0.3–3.6% under P−4. Further investigation shows that in several instances, under high and especially very high penalties, the MDCLP−S improvement phase could not outperform MDCLP−H, hence the latter solution was chosen by MDCLP−S (counting as a zero improvement). This means that exploiting the flexibility introduced by soft unloading constraints is challenging under very high penalties, and suggests that planning based on hard constraints may suffice when the operator weighs heavily the indirect cost of relocating items during delivery. Nevertheless, the statistics in the table support the explicit consideration of soft unloading constraints in general, as they can lead to solutions of much higher objective values compared to directly enforcing hard unloading constraints. Finally, we notice that the improvement is slightly larger for the more homogeneous BR classes, especially under P−3 and P−4.

Table 5 compares MDCLP−B and MDCLP−S. It shows the objective function value of the two strategies under all penalties and BR classes (the results for MDCLP−S are analogous to those in Table 4) and the average improvement by MDCLP−S over MDCLP−B expressed as percentage, denoted Δ_{SB} (%).

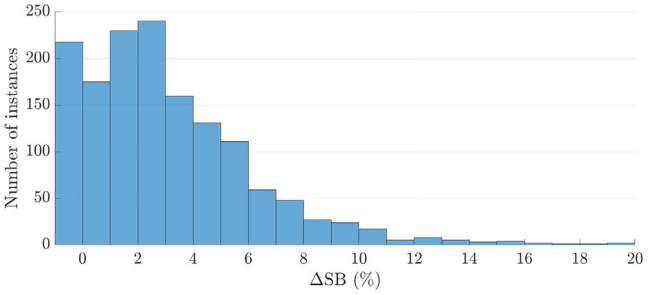


Fig. 2. Distribution of improvements of MDCLP−S over MDCLP−B for medium penalties (P−2).

For all BR classes, the amount of improvement (i.e., Δ_{SB}) increases with the coefficients (α, β) of the penalty function. On average, this improvement varies between 2.4% with P−1 to 10.5% with P−4, reaching a maximum of about 15% for BR5 under P−4. The reason behind this behavior is that MDCLP−S is aware of the penalty when optimizing the trade-off between cargo value and unloading cost in the improvement phase, while MDCLP−B is passively subject to increasing penalties. In other terms, the loading solutions by MDCLP−B are generated during the construction phase independently of the penalty, maximizing the cargo value, hence indirectly the filling degree of the container. These solutions typically present many unloading constraints violations, whose cost can be marginal under low penalties such as P−1 but is substantial under higher penalties like P−3 or P−4. Overall, the numbers in Table 5 reveal that neglecting the unloading constraints during the packing phase leads to poor solutions with a potentially high cargo value but that are very expensive to unload. Overcoming this issue requires applying an improvement phase that accounts for the unloading costs.

Next, we investigate more in depth the benefit of the improvement phase of our heuristic, and provide more insights on the decomposition of the objective function into cargo value and penalties.

5.3. Analysis of improvement

In Fig. 2, we show the distribution over all 1500 instances of the improvement Δ_{SB} under penalty P−2, which was only reported as

Table 5
Comparison of MDCLP-B and MDCLP-S strategies.

	MDCLP-B				MDCLP-S				Δ_{SB} (%)			
Class	P-1	P-2	P-3	P-4	P-1	P-2	P-3	P-4	P-1	P-2	P-3	P-4
BR1	82.0	77.1	74.1	71.5	84.1	81.4	80.4	79.9	2.5	5.6	8.4	11.7
BR2	81.9	77.7	74.5	71.6	83.4	80.6	79.4	78.7	1.8	3.7	6.7	9.9
BR3	79.6	73.8	69.6	65.8	80.9	77.0	75.3	74.5	1.7	4.3	8.1	13.2
BR4	78.7	73.4	69.8	66.6	80.4	76.5	75.0	74.4	2.1	4.2	7.4	11.7
BR5	78.2	72.0	67.7	63.8	79.6	75.6	74.0	73.2	1.8	5.0	9.4	14.9
BR6	77.3	71.8	67.9	64.2	79.0	74.6	72.8	72.0	2.2	3.9	7.4	12.2
BR7	76.3	71.1	67.1	63.5	78.1	73.7	72.3	71.1	2.3	3.7	7.6	12.0
BR8	75.7	70.7	67.3	64.2	77.7	73.4	71.4	70.4	2.7	3.8	6.1	9.7
BR9	75.1	70.6	67.3	64.2	77.0	72.7	70.9	70.0	2.5	3.0	5.3	9.0
BR10	74.4	70.2	66.9	63.8	76.5	72.2	70.4	69.4	2.7	2.9	5.2	8.8
BR11	74.5	70.2	67.1	64.2	76.4	72.4	70.4	69.8	2.6	3.0	4.9	8.8
BR12	74.1	69.6	66.2	62.9	76.1	71.4	69.6	68.8	2.7	2.6	5.1	9.3
BR13	74.4	70.4	67.3	64.3	76.3	72.0	70.0	69.2	2.6	2.3	4.1	7.6
BR14	74.0	69.6	66.1	62.8	75.8	71.4	69.4	68.6	2.4	2.6	5.0	9.2
BR15	73.8	69.5	66.0	62.8	75.9	71.4	69.6	68.9	2.8	2.8	5.3	9.8
Mean	76.7	71.8	68.3	65.1	78.5	74.4	72.7	71.9	2.4	3.5	6.4	10.5

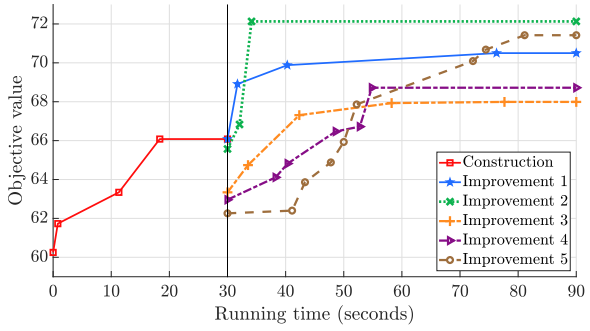


Fig. 3. Evolution of objective function value in 4th BR7 instance under penalty P-2.

an average in Table 5. To ease visualization, we collect around 220 instances with zero improvement ($\Delta_{SB} = 0\%$) into a first bar depicted for range $[-1, 0]$, and truncate the graph at $\Delta_{SB} = 20\%$, discarding about 20 instances with larger improvements.

The distribution is right skewed. Although the bulk of the improvement lies between 0 and 6%, with an average of 3.5%, there are 158 instances (or 10.5%) with improvements of 6–10%, and further 77 instances (or 5.1%) with improvements of more than 10%. This suggests that there is a certain variability in the effectiveness of our improvement heuristic with respect to different packing configurations. Nonetheless, applying this phase is critical to avoid huge losses in a significant subset of instances. This is even more emphasized under higher penalties such as P-3 and P-4 where the proportion of instances with improvements above 10% is roughly 20% and 37%, respectively.

For a single instance, Fig. 3 shows how the objective function value evolves during construction and improvement phases. For this illustration, we extended the improvement phase to 60 seconds per solution and display markers only for the first, last, and improving iterations of both phases.

Construction and improvement iterations both play an essential role in this example as they increase the objective value from about 60 to 66 and from 66 to 72, respectively. Let us denote by $\mathcal{S}_1, \ldots, \mathcal{S}_5$ the five best solutions obtained at the end of the construction phase, in this order. In the instance considered in Fig. 3, it is the second best solution $\mathcal{S}_2$ (dotted green line) that results in the overall best solution after experiencing an improvement of 6.5. Instead, $\mathcal{S}_1$ (continuous blue line) could only be improved by 4.4, ranking third after improvement. Interestingly, $\mathcal{S}_5$ (dashed brown line) could be improved by 9.2, ending up second after improvement. This clearly shows that seemingly worse solutions from the construction phase may embed more potential for improvement.

What we observe here is not an isolated case. We found indeed that $\mathcal{S}_1$ produced the best packing after the improvement phase in about 40% of the instances (including those instances with zero improvement), while the remaining 60% was provided by solutions $\mathcal{S}_2$ to $\mathcal{S}_5$, each with at least a 10% share. This proves that improving an array of different packing solutions is helpful whilst focusing on just a single one may be inefficient. On the BR7 instances, for example, the average improvements Δ_{SB} shown in Table 5 would be roughly halved when only improving $\mathcal{S}_1$. Given this analysis and results, applying the improvement phase to an even larger number of solutions may have the potential to bring some additional gain, at the obvious expense of higher computing time.

We finally investigate whether the improvement observed in Figs. 2, 3 stems from increasing the cargo value, from reducing the unloading costs, or both, and show our results in Table 6. This table contains the cargo value (labeled "Val."), the incurred penalties ("Pen."), and their difference ("Dif."), obtained after construction and improvement phases under all BR classes and penalty sets. Notice that for the construction phase, "Dif." coincides with the MDCLP-B objective, but for the improvement phase, "Dif." differs from the MDCLP-S objective as the latter is lower bounded for each instance by the hard constrained solution. This means that "Dif." for the improvement phase is always less than or equal to the objective of the MDCLP-S heuristic discussed in Section 5.2. Green and red entries of the table highlight, respectively, improvements and worsening in cargo value or penalties.

The results are mixed and depend on both penalty weights and heterogeneity of instance. Under the intermediate penalties P-2 and P-3, the cargo value and the unloading costs often improve simultaneously. Under low penalties (P-1), some extra unloading cost is usually introduced with the goal of increasing the cargo value by a larger extent. Under very high penalties (P-4), the opposite happens and some cargo value is sacrificed during improvement to cut the unloading costs by a more significant amount. Overall, this analysis shows that balancing the trade-off between the two objective function components is crucial to ensure the highest solution quality, but is not straightforward without an improvement phase as it depends on the instance's features and on the penalties.

5.4. Comparison with commercial solver

In this section, we compare the solutions of our heuristic strategy MDCLP-S with those obtained when solving the MILP with a

Table 6
Analysis of improvement: cargo value vs. unloading cost.

	P-1						P-2					
	Construction			Improvement			Construction			Improvement		
Class	Val.	Pen.	Dif.	Val.	Pen.	Dif.	Val.	Pen.	Dif.	Val.	Pen.	Dif.
BR1	85.4	3.4	82.0	86.7	3.0	83.7	82.6	5.5	77.1	83.8	4.1	79.8
BR2	85.0	3.1	81.9	86.1	2.9	83.2	83.0	5.3	77.7	83.7	4.0	79.7
BR3	83.6	4.0	79.6	85.0	4.1	80.9	80.8	7.0	73.8	81.7	5.6	76.1
BR4	82.6	3.9	78.7	84.5	4.2	80.3	79.7	6.3	73.4	81.3	5.9	75.4
BR5	82.8	4.6	78.2	84.2	4.6	79.6	79.2	7.2	72.0	80.9	6.3	74.6
BR6	81.3	4.0	77.3	83.4	4.4	79.0	78.2	6.4	71.8	80.4	6.2	74.2
BR7	80.4	4.1	76.3	82.6	4.5	78.1	77.4	6.3	71.1	79.1	6.1	73.0
BR8	79.8	4.1	75.7	82.2	4.5	77.7	76.2	5.5	70.7	78.7	5.7	73.1
BR9	78.8	3.7	75.1	81.6	4.6	77.0	75.7	5.1	70.6	78.2	5.7	72.5
BR10	78.4	4.0	74.4	81.1	4.6	76.5	75.3	5.1	70.2	77.8	5.7	72.1
BR11	78.6	4.1	74.5	80.8	4.4	76.4	75.2	5.0	70.2	78.0	5.9	72.1
BR12	78.2	4.1	74.1	80.8	4.7	76.1	75.0	5.4	69.6	77.6	6.3	71.3
BR13	77.6	3.2	74.4	80.7	4.4	76.3	75.5	5.1	70.4	77.5	5.8	71.8
BR14	77.9	3.9	74.0	80.5	4.6	75.8	75.0	5.4	69.6	77.3	6.2	71.2
BR15	77.7	3.9	73.8	80.5	4.6	75.9	75.0	5.5	69.5	77.2	6.1	71.2
Mean	80.6	3.9	76.7	82.7	4.3	78.4	77.5	5.7	71.8	79.5	5.7	73.9
	P-3						P-4					
	Construction			Improvement			Construction			Improvement		
Class	Val.	Pen.	Dif.	Val.	Pen.	Dif.	Val.	Pen.	Dif.	Val.	Pen.	Dif.
BR1	81.8	7.7	74.1	82.1	3.9	78.2	80.5	9.0	71.5	80.6	3.0	77.6
BR2	82.0	7.5	74.5	81.6	4.1	77.5	81.5	9.9	71.6	79.9	4.0	75.9
BR3	79.3	9.7	69.6	79.6	6.6	73.0	78.6	12.8	65.8	77.2	6.3	71.0
BR4	78.1	8.3	69.8	78.7	6.1	72.6	77.9	11.3	66.6	76.8	5.9	70.9
BR5	77.8	10.1	67.7	77.7	6.1	71.7	77.2	13.4	63.8	75.1	5.4	69.7
BR6	77.4	9.5	67.9	78.0	6.6	71.4	76.9	12.7	64.2	75.7	6.5	69.3
BR7	76.5	9.4	67.1	77.0	6.7	70.4	76.1	12.6	63.5	74.8	7.2	67.6
BR8	75.4	8.1	67.3	76.7	6.5	70.2	74.9	10.7	64.2	74.5	6.7	67.8
BR9	75.1	7.8	67.3	76.4	6.5	69.9	74.8	10.6	64.2	74.3	6.7	67.6
BR10	74.8	7.9	66.9	75.8	6.7	69.1	74.4	10.6	63.8	73.8	7.1	66.7
BR11	74.5	7.4	67.1	75.7	6.7	69.1	74.4	10.2	64.2	73.9	7.2	66.7
BR12	74.5	8.3	66.2	75.7	7.6	68.1	74.1	11.2	62.9	73.4	7.9	65.5
BR13	74.8	7.5	67.3	75.7	6.7	69.0	74.5	10.2	64.3	73.8	7.4	66.4
BR14	74.5	8.4	66.1	75.5	7.4	68.0	74.2	11.4	62.8	73.2	8.1	65.1
BR15	74.4	8.4	66.0	75.3	7.3	68.0	74.1	11.3	62.8	72.9	7.8	65.1
Mean	76.7	8.4	68.3	77.4	6.4	71.1	76.3	11.2	65.1	75.3	6.5	68.9

Table 7
Comparison of MDCLP-S and Gurobi on BR4 and BR10 instances.

		MDCLP-S	Gurobi				MDCLP-S	Gurobi	
BR4	Items	130s	130s	600s	BR10	Items	130s	130s	600s
1	106	79.56	18.92	22.56	1	136	76.90	3.11	3.11
2	123	79.16	6.69	6.69	2	123	75.80	20.66	20.66
3	135	81.69	5.57	5.57	3	136	71.52	2.93	2.93
4	169	77.85	3.05	3.05	4	122	78.71	3.71	3.71
5	130	80.50	9.12	9.12	5	133	72.84	3.75	3.75
6	132	81.89	2.89	56.78	6	159	71.62	1.88	1.88
7	138	82.61	6.28	6.28	7	127	74.63	4.47	4.47
8	107	74.94	16.74	27.10	8	123	72.03	6.63	6.63
9	149	78.04	2.36	4.92	9	107	72.05	3.72	7.22
10	133	79.69	3.91	3.91	10	126	72.16	6.27	6.27
Mean	132	79.59	7.55	14.60	Mean	129	73.83	5.71	6.06

commercial optimization solver, namely Gurobi 9.1. We recall that the complete MILP is reported in the appendix in (11). We performed two sets of experiments: (i) using directly some of the BR instances, to investigate the limits of the solver, and (ii) on smaller instances that can be solved to optimality, so that we can compare our results with optimal solutions.

In the first experiment, we consider 20 BR instances from classes BR4 and BR10 (to include different degrees of heterogeneity) and solve them with penalty P-3, excluding the reachability constraints (10c) which are not modeled in the MILP. Table 7 shows the objective function value achieved by MDCLP-S running 130 seconds (of which 30 for construction and 100 to improve 5 solutions) and from Gurobi with the same time budget as well as an increased budget of 600 seconds.

It is evident from the table that the solver struggles to find good feasible solutions to instances with over 100 items. In most cases, the solver does not retrieve any improving solution when moving from 130 to 600 seconds, and the values are often so low that the best solutions found are essentially useless. This strengthens the discussion in Section 3.3 about the need for developing a heuristic method.

We verified that moderate-sized instances of 40–60 items are also intractable. Thus, we need to scale down the instance size considerably in order to compare our results with optimal solu-

Table 8
Comparison of MDCLP−S and Gurobi on reduced BR4 instances.

	Gurobi					MDCLP−S				
BR4	Val.	Pen.	Obj.	Items	Time	Val.	Pen.	Obj.	Items	Gap (%)
1	89.99	0.00	89.99	11	244.8	87.61	2.97	84.65	10	5.9
2	84.33	0.38	83.95	11	9.5	73.95	1.52	72.43	10	13.7
3	92.31	1.08	91.22	10	922.5	87.38	0.82	86.56	10	5.1
4	85.59	0.73	84.86	11	9.4	74.52	2.84	71.68	10	15.5
5	92.05	1.22	90.83	10	689.2	88.56	0.87	87.69	9	3.5
6	85.21	0.77	84.44	10	>7200.0	85.72	5.31	80.41	10	4.8
7	89.12	0.52	88.60	11	40.5	81.47	2.98	78.49	10	11.4
8	90.52	0.50	90.02	11	28.7	86.40	1.79	84.61	10	6.0
9	85.55	1.77	83.78	10	318.2	76.29	2.15	74.14	10	11.5
10	82.57	0.00	82.57	10	5.1	78.97	0.00	78.97	9	4.4
Mean	87.72	0.70	87.03	10.5	946.8	82.09	2.12	79.96	9.8	8.1

tions. In the following, we consider 10 instances of 12 items constructed from the BR4 instances previously used in Table 7. More specifically, we choose the 12 items so that the proportion of demands across item types is respected as much as possible (e.g., an instance with 100 and 50 items of type 1 and 2, respectively, would be turned into a 12-item instance with 8 and 4 items of type 1 and 2, respectively). Moreover, the dimensions of all boxes are scaled up so that the joint volume of the 12 boxes becomes 100% the volume of the container. For these instances, the results in Table 8 show the objective function value obtained by MDCLP−S and Gurobi, its breakdown into cargo value and penalty, the number of loaded items, and the runtime of the solver capped at 2 hours.

Our heuristic solutions are on average 8% away from the optimal. The number of loaded items suggests that a reason for this gap is the ability of the solver to better fill the volume of the container by loading one more box on average (or loading larger boxes), which in small instances translates to a significant proportion of cargo value. Although the penalties in the optimal solutions improve too compared to our heuristic, their contribution to the gap is only about 1.5%. The computational time required by the solver fluctuates significantly, which is impractical. Whilst some instances were solved in ten seconds, others needed several hundred seconds, and one was not solved to optimality in 2 hours. Thus, although there may be a benefit in using off-the-shelf commercial optimization software, this benefit is limited to problems that are extremely limited in size.

5.5. Comparison with methods from the literature

The approach in the literature that is closest to ours is Gajda et al. (2022), where penalties are item-independent, and each box violating the unloading constraints counts as one (see Section 2). The solution method developed in Gajda et al. (2022) is a randomized constructive heuristic called RCH. Here, we compare RCH with our MDCLP−S heuristic to investigate whether considering the number of boxes to relocate may be a sufficient approach for the MDCLP with soft unloading constraints.

Since RCH accounts for a variety of practical constraints that we do not consider (e.g., vertical stability, loading priorities, and weight distribution), a direct comparison would be unfair for RCH. Thus, we implemented a simplified version of RCH that neglects such additional practical constraints, and tested it for all 1500 BR instances and penalties P−1, P−2, P−3, and P−4. Table 9 reports the averaged RCH results and their percentage difference with respect to MDCLP−S (see Table 6). Both methods have a time limit of 130 seconds. Unlike MDCLP−S, notice that RCH does not have an improvement phase; hence, the time budget available is used entirely for the construction phase.

When looking at the cargo value alone, RCH slightly outperforms MDCLP−S, especially for the more heterogeneous instances, and by 1.2% on average across all BR classes and penalties. In contrast, RCH incurs penalties that are substantially higher and up to three times those incurred by MDCLP−S. On average across all BR classes, the penalized value achieved by RCH is 0.5%, 4%, 7.8%, and 12% worse than MDCLP−S under penalty P−1, P−2, P−3, and P−4, respectively. These statistics reveal that if penalties are very small (e.g., P−1), then putting more emphasis on maximizing cargo value may be a good strategy, whereas incorporating advanced techniques to reduce penalties such as region reconstruction leads to limited gains. For example, RCH uses a preprocessing phase during construction to combine items into blocks, which possibly explains the slight improvement in cargo value. Nevertheless, if penalties are more significant (e.g., P−2 to P−4) and item-specific, then simply targeting a reduction in the number of item relocations is inefficient as shown by the large performance gap.

Besides considering the objective function used in this paper, to improve fairness we also compare the performance of RCH and MDCLP−S on the trade-off between cargo value and the number of items to relocate, managing which is a goal in Gajda et al. (2022). To this end, we show in Fig. 4 the randomized RCH and MDCLP−S iterations and their implied Pareto frontiers on three BR instances. To improve visualization, we have truncated some less interesting portions of the solution space.

As we can see, results are mixed and no method dominates the other, with the two Pareto frontiers often crossing each other. In conclusion, our approach to tackle soft unloading constraints implicitly reduces the number of items to relocate as effectively as RCH, in addition to dominating this method considerably when it comes to minimizing the penalties.

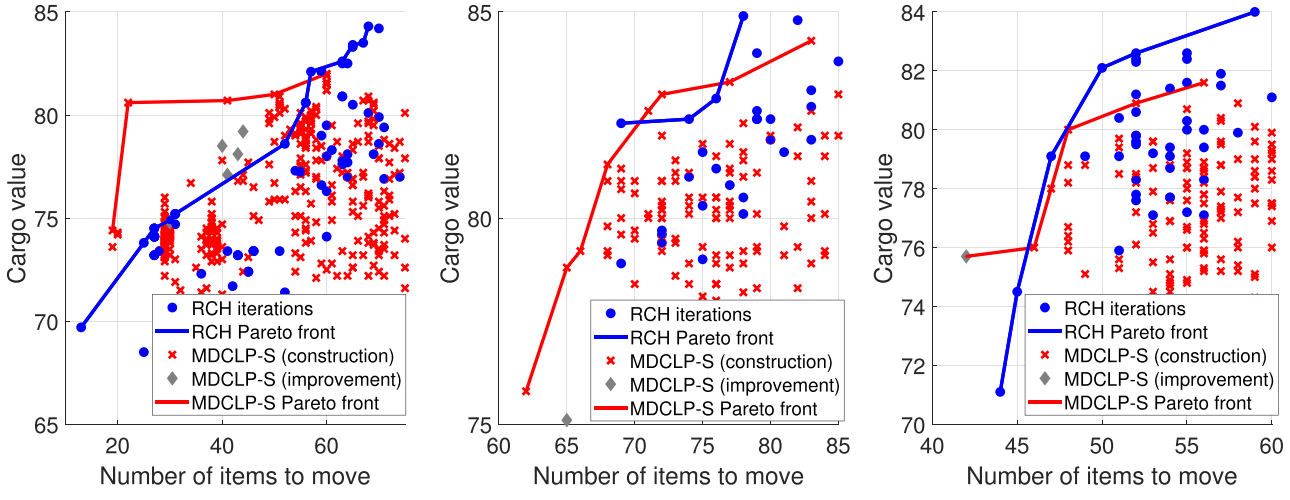
5.6. Region reconstruction methods

Finally, we compare the three methods introduced in Section 4.2 to define the region to reconstruct during the improvement phase, namely M−1, M−2, and M−3 (see Step 2 of Algorithm 1). Our setting involves $\xi = 5$ in all methods and trade-off parameters $\zeta^{\mathrm{D}} = 0.7$ and $\zeta^{\mathrm{E}} = 0.3$ in M−3, which we picked based on preliminary experiments. In Table 10, we report the average objective value obtained when employing these three methods for each BR class and parameter set. To ensure consistency, for each instance we improve the same five constructive solutions when applying M−1, M−2, and M−3.

It is evident that approach M−1 based on the conflict matrix outperforms the others in most instances, except for the more homogeneous classes BR1–5 under penalty P−1 where instead M−2 and M−3 produce slightly better results. Typically, the performance difference between M−1 and the best among M−2 and M−3 is marginal but can also reach up to 1–2% in some instances. Recall

Table 9
RCH results on the BR instances and comparison with MDCLP-S.

	P-1			P-2			P-3			P-4		
Class	Val.	Pen.	Obj.	Val.	Pen.	Obj.	Val.	Pen.	Obj.	Val.	Pen.	Obj.
BR1	84.0	2.8	81.2	82.0	5.0	77.0	80.4	6.3	74.2	79.0	7.1	72.0
	-3.1%	-6.3%	-3.0%	-2.2%	+ 21%	-3.5%	-2.1%	+ 60%	-5.2%	-1.9%	+ 136%	-7.3%
BR2	83.1	3.8	79.3	80.6	6.5	74.1	78.5	8.2	70.3	77.9	10.8	67.2
	-3.5%	+ 31%	-4.7%	-3.7%	+ 62%	-7.1%	-3.8%	+ 100%	-9.3%	-2.5%	+ 169%	-11.5%
BR3	83.3	5.3	78.1	78.9	8.0	70.9	77.3	10.9	66.4	76.2	14.0	62.2
	-2.0%	+ 28%	-3.5%	-3.5%	+ 42%	-6.8%	-2.9%	+ 65%	-9.1%	-1.3%	+ 122%	-12.4%
BR4	83.6	5.7	77.8	79.0	8.6	70.4	76.1	10.8	65.3	75.3	14.1	61.1
	-1.1%	+ 36%	-3.1%	-2.9%	+ 46%	-6.7%	-3.3%	+ 76%	-10.0%	-2.0%	+ 139%	-13.8%
BR5	83.2	5.1	78.1	79.7	8.8	71.0	78.3	12.7	65.6	77.3	16.6	60.7
	-1.2%	+ 12%	-1.9%	-1.4%	+ 39%	-4.9%	+ 0.8%	+ 107%	-8.5%	+ 2.9%	+ 207%	-12.9%
BR6	83.2	6.0	77.1	78.4	8.9	69.6	77.3	13.3	64.0	76.2	17.3	58.9
	-0.3%	+ 37%	-2.4%	-2.4%	+ 43%	-6.3%	-0.9%	+ 102%	-10.4%	+ 0.6%	+ 165%	-15.0%
BR7	83.1	5.6	77.6	78.9	8.7	70.2	77.8	13.1	64.7	77.3	17.7	59.6
	+ 0.6%	+ 24%	-0.7%	-0.2%	+ 43%	-3.9%	+ 1.1%	+ 95%	-8.0%	+ 3.3%	+ 146%	-11.8%
BR8	83.7	5.4	78.3	80.2	9.6	70.6	79.3	14.9	64.4	78.1	19.5	58.7
	+ 1.8%	+ 21%	+ 0.7%	+ 1.9%	+ 69%	-3.4%	+ 3.4%	+ 129%	-8.2%	+ 4.9%	+ 191%	-13.5%
BR9	84.1	6.2	77.9	79.2	9.1	70.1	77.8	13.4	64.4	77.2	18.0	59.2
	+ 3.0%	+ 35%	+ 1.1%	+ 1.2%	+ 60%	-3.4%	+ 1.8%	+ 106%	-7.8%	+ 3.8%	+ 168%	-12.5%
BR10	84.0	5.9	78.1	79.4	8.8	70.7	78.6	13.5	65.1	77.4	17.6	59.9
	+ 3.6%	+ 28%	+ 2.1%	+ 2.1%	+ 54%	-2.0%	+ 3.6%	+ 101%	-5.8%	+ 4.9%	+ 147%	-10.2%
BR11	84.5	6.0	78.4	80.0	9.1	70.9	78.7	13.5	65.2	78.5	18.7	59.9
	+ 4.5%	+ 37%	+ 2.7%	+ 2.6%	+ 54%	-1.6%	+ 4.0%	+ 102%	-5.6%	+ 6.2%	+ 159%	-10.3%
BR12	84.2	6.5	77.8	79.6	10.0	69.6	78.8	15.6	63.2	77.9	20.8	57.1
	+ 4.2%	+ 38%	+ 2.2%	+ 2.6%	+ 58%	-2.4%	+ 4.1%	+ 105%	-7.2%	+ 6.1%	+ 163%	-12.8%
BR13	83.8	6.1	77.7	79.3	8.8	70.4	78.8	14.1	64.7	78.2	19.0	59.2
	+ 3.8%	+ 38%	+ 1.8%	+ 2.3%	+ 52%	-1.9%	+ 4.1%	+ 110%	-6.2%	+ 6.0%	+ 157%	-10.8%
BR14	83.0	6.7	76.3	78.2	9.6	68.6	77.4	15.0	62.4	76.5	20.0	56.5
	+ 3.0%	+ 45%	+ 0.6%	+ 1.1%	+ 54%	-3.7%	+ 2.5%	+ 103%	-8.2%	+ 4.5%	+ 146%	-13.2%
BR15	83.0	6.4	76.6	78.2	9.2	69.0	77.5	14.5	63.1	77.0	19.6	57.4
	+ 3.1%	+ 38%	+ 0.9%	+ 1.3%	+ 51%	-3.1%	+ 2.9%	+ 98%	-7.3%	+ 5.6%	+ 151%	-11.8%
Mean	83.6	5.6	78.0	79.4	8.6	70.9	78.2	12.6	65.5	77.3	16.7	60.6
	+ 1.1%	+ 29%	-0.5%	-0.1%	+ 50%	-4.0%	+ 1.0%	+ 97%	-7.8%	+ 2.7%	+ 158%	-12.0%



(a) BR4 instance 3. (b) BR10 instance 5. (c) BR10 instance 7.

Fig. 4. Trade-off between cargo value and item relocations under RCH and MDCLP-S.

that M-2 chooses the region based on its penalty density, which encodes information on conflicts like M-1 as well as additional information on the volume where conflicts occur. Thus, the superior performance of M-1 over M-2 was unexpected, but we attribute it to the latter approach favoring the selection of smaller regions that are more likely to exhibit a high penalty density. Considering such small regions may be less efficient as they provide less flexibility for reconstruction. Also surprising is the contrast between M-2 and M-3 since the latter method exploits extra information on empty spaces but performs better in less than half of the instances (mostly those with high penalties). Therefore, our results suggest that empty spaces and penalty densities do not convey critical information to select the region to reconstruct, while letting conflicts guiding the search seems sufficient.

Finally, Table 11 shows the number of "wins" each region definition method achieves over the 100 instances of each class, and complements the former table that only reports average performance. In case of ties among two or three methods, we attribute one point to each of them, meaning that the total score may add up to more than 100. The results from this table are consistent with the previous discussion since M-1 obtains the highest score in most configurations of BR classes and penalties,

Table 10
Comparison of region reconstruction methods: objective function value.

	P-1			P-2			P-3			P-4		
Class	M-1	M-2	M-3	M-1	M-2	M-3	M-1	M-2	M-3	M-1	M-2	M-3
BR1	83.7	83.6	**83.8**	79.8	79.3	**79.9**	**78.2**	76.2	78.1	**77.6**	74.4	**77.6**
BR2	**83.2**	**83.2**	**83.2**	**79.7**	79.2	78.9	**77.5**	76.1	77.2	**75.9**	73.4	75.3
BR3	80.9	**81.0**	80.9	**76.1**	75.6	75.7	**73.0**	71.4	72.4	**71.0**	68.0	70.2
BR4	80.3	**80.4**	80.2	**75.4**	75.1	75.0	**72.6**	71.6	72.5	**70.9**	68.3	70.4
BR5	79.6	**79.7**	79.6	**74.6**	73.7	73.6	**71.7**	69.8	70.8	**69.7**	66.4	68.8
BR6	**79.0**	78.8	78.7	**74.2**	73.5	73.6	**71.4**	69.8	70.5	**69.3**	67.3	68.4
BR7	**78.1**	78.0	77.7	**73.0**	72.7	72.4	**70.4**	69.0	69.4	**67.6**	65.5	66.9
BR8	**77.7**	77.5	77.1	**73.1**	72.5	71.9	**70.2**	69.3	69.0	**67.8**	66.2	66.7
BR9	**77.0**	76.9	76.4	**72.5**	72.2	71.6	**69.9**	68.9	68.8	**67.6**	66.1	66.3
BR10	**76.5**	76.2	75.7	**72.1**	71.7	71.0	**69.1**	68.4	68.2	**66.7**	65.3	65.7
BR11	**76.4**	76.3	75.8	**72.1**	71.6	71.3	**69.1**	68.5	68.3	**66.7**	65.8	65.7
BR12	**76.1**	75.9	75.5	**71.3**	70.9	70.4	**68.1**	67.6	67.2	**65.5**	64.5	64.6
BR13	**76.3**	76.1	75.5	**71.8**	71.7	71.1	**69.0**	68.4	68.1	**66.4**	65.4	65.6
BR14	**75.8**	75.7	75.2	**71.2**	70.8	70.2	**68.0**	67.3	66.8	**65.1**	64.4	64.0
BR15	**75.9**	75.5	74.8	**71.2**	70.8	70.1	**68.0**	67.3	66.8	**65.1**	64.2	64.5
Mean	**78.4**	78.3	78.0	**73.9**	73.4	73.1	**71.1**	70.0	70.3	**68.9**	67.0	68.0

Table 11
Comparison of region reconstruction methods: number of wins.

	P-1			P-2			P-3			P-4		
Class	M-1	M-2	M-3	M-1	M-2	M-3	M-1	M-2	M-3	M-1	M-2	M-3
BR1	30	**46**	39	39	49	**52**	49	**58**	56	53	57	**58**
BR2	**35**	34	34	41	**50**	29	**47**	**47**	41	**61**	44	51
BR3	24	**46**	34	41	**43**	35	**54**	44	36	**61**	38	41
BR4	32	**40**	34	40	**41**	33	**45**	32	39	**56**	30	36
BR5	27	**42**	31	**53**	28	25	**49**	34	31	**56**	37	35
BR6	**45**	40	23	**51**	29	28	**59**	28	31	**56**	28	32
BR7	**44**	36	25	40	**44**	26	**53**	30	25	**55**	27	36
BR8	**48**	31	21	**61**	33	20	**61**	32	24	**59**	31	33
BR9	**46**	34	22	**57**	38	21	**69**	29	19	**66**	31	26
BR10	**47**	41	12	**63**	38	11	**62**	30	24	**66**	25	29
BR11	**48**	38	16	**61**	27	22	**52**	30	26	**60**	33	25
BR12	**48**	40	14	**56**	36	22	**57**	44	21	**63**	26	28
BR13	**43**	39	18	**44**	42	24	**58**	38	20	**61**	30	35
BR14	**42**	**42**	20	**54**	38	17	**60**	28	22	**60**	35	19
BR15	**53**	37	13	**55**	35	18	**65**	34	19	**58**	29	33
Mean	**40.8**	39.1	23.7	**50.4**	38.1	25.5	**56.0**	35.9	28.9	**59.4**	33.4	34.5

emerging as the most robust method to select a region. However, M-1 rarely exceeds a score of 55 or 60, implying that M-2 and M-3 still win in a significant subset of instances. E.g., M-2 is comparable to M-1 for low penalties and weakly-heterogeneous instances. This hints that extending our algorithm by randomizing the region reconstruction method (i.e., picking a different method at each improvement iteration) may have some potential to further improve a packing solution.

6. Conclusion

The aim of this study was to model and solve a variant of the MDCLP in which the unloading constraints are not enforced as hard constraints, as commonly done in the literature, but are instead treated as soft constraints, which is relevant in practice for logistics operators (see, e.g., the discussion in Gajda et al., 2022). To this end, we introduced penalty functions that are activated when above, visibility, and reachability unloading constraints are violated between item pairs belonging to different customers. The definition of these penalties is flexible and allows an operator to model the indirect cost/time of moving items during delivery (emphasizing, e.g., heavy or bulky items) and consequently account for the trade-off between this cost and the value of the transported cargo.

After presenting a mixed-integer programming formulation able to tackle small instances under specific penalties, we proposed a more general heuristic framework made of fast construction and improvement phases, and tested it on a large set of instances from the literature. Our computational study provides a set of algorithmic insights (e.g., on the role of construction vs. improvement, on the region reconstruction techniques, and on the limited usefulness of commercial optimization solvers) as well as managerial insights. In particular, it shows that a loading strategy incorporating soft unloading constraints may be significantly more efficient than a sequential approach that only evaluates penalties a posteriori, a hard unloading constraint approach, and a heuristic from the literature that counts the number of box relocations without directly minimizing the penalties. Only under high penalties, planning based on hard unloading constraints may be enough.

Future research may be targeted at improving the heuristic algorithm by: (i) adding movements to the improvement phase from those presented in the VNS of Parreño et al. (2010), e.g., layer reduction, column insertion, and swapping of items or groups of items, (ii) trying alternative merit functions in *Step 4* of Algorithm 1, or (iii) using the graph representation of Fekete et al. (2007) not only to shift boxes and obtain gapless solutions, but also to modify the relative position of items by varying the transitive orientations of the interval graphs as done in Trivella & Pisinger (2016).

The problem we consider is static and all penalties are computed based on the initial packing configuration. In practice, the multi-drop process may also allow at each delivery point to reposition a portion of the cargo inside the container, potentially eliminating some violations of the unloading constraints for later customers or creating new ones. Accounting for the extra decisions

to where to relocate items during the multi-delivery phase constitutes an additional layer of flexibility that would be interesting to investigate in a dynamic (and certainly more complex) MDCLP extension.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is partly supported by the Swiss National Science Foundation under Project 1481210/DADA.

Appendix

In this appendix, we present an MILP formulation for the MDCLP with soft unloading constraints and that includes rotations. Model (11) relies on the same notation as Section 3, and in addition, employs binary variables $o_{i1}, \ldots, o_{i6} \in \{0, 1\}$ for $i \in \mathcal{B}$ to model the six different orthogonal orientations of a cuboid. Constraints (11i) ensure that only one such orientation is selected for each item $i \in \mathcal{B}$.

$$\max \quad \sum_{i\in\mathcal{B}} \pi_i t_i - \sum_{i\in\mathcal{B}} \sum_{j\in\mathcal{B}:c_i<c_j} \left[p_{ij}\left(\alpha_{\text{A}} q_j + \beta_{\text{A}} v_j + \gamma_{\text{A}} z_j + \eta_{\text{A}}\right) + r_{ij}\left(\alpha_{\text{v}} q_j + \beta_{\text{v}} v_j + \gamma_{\text{v}} z_j + \eta_{\text{v}}\right)\right] \tag{11a}$$

$$\text{s.t.:} \quad f_{ij} + f_{ji} + b_{ij} + b_{ji} + u_{ij} + u_{ji} + (1 - t_i) + (1 - t_j) \geq 1 \quad \forall i, j \in \mathcal{B},\ i < j, \tag{11b}$$

$$x_i + l_i(o_{i1} + o_{i2}) + w_i(o_{i3} + o_{i4}) + h_i(o_{i5} + o_{i6}) - x_j \leq L(1 - b_{ij}) \quad \forall i, j \in \mathcal{B}, \tag{11c}$$

$$y_i + l_i(o_{i3} + o_{i5}) + w_i(o_{i1} + o_{i6}) + h_i(o_{i2} + o_{i4}) - y_j \leq W(1 - f_{ij}) \quad \forall i, j \in \mathcal{B}, \tag{11d}$$

$$z_i + l_i(o_{i4} + o_{i6}) + w_i(o_{i2} + o_{i5}) + h_i(o_{i1} + o_{i3}) - z_j \leq H(1 - u_{ij}) \quad \forall i, j \in \mathcal{B}, \tag{11e}$$

$$x_i + l_i(o_{i1} + o_{i2}) + w_i(o_{i3} + o_{i4}) + h_i(o_{i5} + o_{i6}) \leq L \quad \forall i, j \in \mathcal{B}, \tag{11f}$$

$$y_i + l_i(o_{i3} + o_{i5}) + w_i(o_{i1} + o_{i6}) + h_i(o_{i2} + o_{i4}) \leq W \quad \forall i, j \in \mathcal{B}, \tag{11g}$$

$$z_i + l_i(o_{i4} + o_{i6}) + w_i(o_{i2} + o_{i5}) + h_i(o_{i1} + o_{i3}) \leq H \quad \forall i, j \in \mathcal{B}, \tag{11h}$$

$$o_{i1} + o_{i2} + o_{i3} + o_{i4} + o_{i5} + o_{i6} = 1 \quad \forall i, j \in \mathcal{B}, \tag{11i}$$

$$x_j \leq x_i + l_i(o_{i1} + o_{i2}) + w_i(o_{i3} + o_{i4}) + h_i(o_{i5} + o_{i6}) + L\, b_{ij} \quad \forall i, j \in \mathcal{B},\ c_i \neq c_j, \tag{11j}$$

$$y_j \leq y_i + l_i(o_{i3} + o_{i5}) + w_i(o_{i1} + o_{i6}) + h_i(o_{i2} + o_{i4}) + W\, f_{ij} \quad \forall i, j \in \mathcal{B},\ c_i \neq c_j, \tag{11k}$$

$$z_j \leq z_i + l_i(o_{i4} + o_{i6}) + w_i(o_{i2} + o_{i5}) + h_i(o_{i1} + o_{i3}) + H\, u_{ij} \quad \forall i, j \in \mathcal{B},\ c_i \neq c_j, \tag{11l}$$

$$b_{ij} + b_{ji} + f_{ij} + f_{ji} \geq 1 - a_{ij} \quad \forall i, j \in \mathcal{B},\ c_i < c_j, \tag{11m}$$

$$b_{ij} + b_{ji} + f_{ij} + f_{ji} \leq 2(1 - a_{ij}) \quad \forall i, j \in \mathcal{B},\ c_i < c_j, \tag{11n}$$

$$f_{ij} + f_{ji} + u_{ij} + u_{ji} \geq 1 - d_{ij} \quad \forall i, j \in \mathcal{B},\ c_i < c_j, \tag{11o}$$

$$f_{ij} + f_{ji} + u_{ij} + u_{ji} \leq 2(1 - d_{ij}) \quad \forall i, j \in \mathcal{B},\ c_i < c_j, \tag{11p}$$

$$p_{ij} + 1 \geq a_{ij} + u_{ij} \quad \forall i, j \in \mathcal{B},\ c_i < c_j, \tag{11q}$$

$$r_{ij} + 1 \geq d_{ij} + b_{ij} \quad \forall i, j \in \mathcal{B},\ c_i < c_j, \tag{11r}$$

$$\text{var.}: \ b_{ij},\ f_{ij},\ u_{ij},\ a_{ij},\ d_{ij},\ p_{ij},\ r_{ij} \in \{0, 1\} \quad \forall i, j \in \mathcal{B}, \tag{11s}$$

$$t_i,\ o_{i1},\ o_{i2},\ o_{i3},\ o_{i4},\ o_{i5},\ o_{i6} \in \{0, 1\} \quad \forall i \in \mathcal{B}, \tag{11t}$$

$$x_i,\ y_i,\ z_i \geq 0 \quad \forall i \in \mathcal{B}. \tag{11u}$$

References

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

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

Bischoff, E. E., Janetz, F., & Ratcliff, M. (1995). Loading pallets with non-identical items. *European Journal of Operational Research, 84*(3), 681–692. https://doi.org/10.1016/0377-2217(95)00031-K.

Bortfeldt, A. (2012). A hybrid algorithm for the capacitated vehicle routing problem with three-dimensional loading constraints. *Computers & Operations Research, 39*(9), 2248–2257. https://doi.org/10.1016/j.cor.2011.11.008.

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

Chen, C. S., Lee, S.-M., & Shen, Q. S. (1995). An analytical model for the container loading problem. *European Journal of Operational Research, 80*(1), 68–76. https://doi.org/10.1016/0377-2217(94)00002-T.

Christensen, S. G., & Rousœ, D. M. (2009). Container loading with multi-drop constraints. *International Transactions in Operational Research, 16*(6), 727–743. https://doi.org/10.1111/j.1475-3995.2009.00714.x.

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

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

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

Fekete, S. P., Schepers, J., & Van der Veen, J. C. (2007). An exact algorithm for higher-dimensional orthogonal packing. *Operations Research, 55*(3), 569–587. https://doi.org/10.1287/opre.1060.0369.

Ferreira, K. M., de Queiroz, T. A., & Toledo, F. M. B. (2021). An exact approach for the green vehicle routing problem with two-dimensional loading constraints and split delivery. *Computers & Operations Research*, 105452. https://doi.org/10.1016/j.cor.2021.105452.

Fuellerer, G., Doerner, K. F., Hartl, R. F., & Iori, M. (2009). Ant colony optimization for the two-dimensional loading vehicle routing problem. *Computers & Operations Research, 36*(3), 655–673. https://doi.org/10.1016/j.cor.2007.10.021.

Fuellerer, G., Doerner, K. F., Hartl, R. F., & Iori, M. (2010). Metaheuristics for vehicle routing problems with three-dimensional loading constraints. *European Journal of Operational Research, 201*(3), 751–759. https://doi.org/10.1016/j.ejor.2009.03.046.

Gajda, M., Trivella, A., Mansini, R., & Pisinger, D. (2022). An optimization approach for a complex real-life container loading problem. *Omega, 107*, 102559. https://doi.org/10.1016/j.omega.2021.102559.

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

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

Hokama, P., Miyazawa, F. K., & Xavier, E. C. (2016). A branch-and-cut approach for the vehicle routing problem with loading constraints. *Expert Systems with Applications, 47*, 1–13. https://doi.org/10.1016/j.eswa.2015.10.013.

Iori, M., Locatelli, M., Moreira, M. C. O., & Silveira, T. (2020). Reactive GRASP-based algorithm for pallet building problem with visibility and contiguity constraints. In *Proceedings of the international conference on computational logistics* (pp. 651–665). Springer. https://doi.org/10.1007/978-3-030-59747-4_42.

Iori, M., & Martello, S. (2010). Routing problems with loading constraints. *Top, 18*(1), 4–27. https://doi.org/10.1007/s11750-010-0144-x.

Iori, M., Salazar-González, J.-J., & Vigo, D. (2007). An exact approach for the vehicle routing problem with two-dimensional loading constraints. *Transportation science, 41*(2), 253–264. https://doi.org/10.1287/trsc.1060.0165.

Junqueira, L., Morabito, R., & Sato Yamashita, D. (2012a). Mip-based approaches for the container loading problem with multi-drop constraints. *Annals of Operations Research, 199*(1), 51–75. https://doi.org/10.1007/s10479-011-0942-z.

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

Lai, K. K., Xue, J., & Xu, B. (1998). Container packing in a multi-customer delivering operation. *Computers & Industrial Engineering, 35*(1–2), 323–326. https://doi.org/10.1016/S0360-8352(98)00085-0.

Laporte, G. (2009). Fifty years of vehicle routing. *Transportation Science, 43*(4), 408–416. https://doi.org/10.1287/trsc.1090.0301.

Liu, J., Yue, Y., Dong, Z., Maple, C., & Keech, M. (2011). On the three-dimensional container packing problem under home delivery service. *Asia-Pacific Journal of Operational Research, 28*(05), 601–621. https://doi.org/10.1142/S0217595911003466.

Lurkin, V., & Schyns, M. (2015). The airline container loading problem with pickup and delivery. *European Journal of Operational Research, 244*(3), 955–965. https://doi.org/10.1016/j.ejor.2015.02.027.

Männel, D., & Bortfeldt, A. (2016). A hybrid algorithm for the vehicle routing problem with pickup and delivery and three-dimensional loading constraints. *European Journal of Operational Research, 254*(3), 840–858. https://doi.org/10.1016/j.ejor.2016.04.016.

Männel, D., & Bortfeldt, A. (2018). Solving the pickup and delivery problem with three-dimensional loading constraints and reloading ban. *European Journal of Operational Research, 264*(1), 119–137. https://doi.org/10.1016/j.ejor.2017.05.034.

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

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

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

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

Pan, L., Chu, S. C. K., Han, G., & Huang, J. Z. (2009). A tree-based wall-building algorithm for solving container loading problem with multi-drop constraints. In *Proceedings of the IEEE international conference on industrial engineering and engineering management* (pp. 538–542). IEEE. https://doi.org/10.1109/IEEM.2009.5373282.

Parreño, F., Alvarez-Valdes, R., Oliveira, J. F., & Tamarit, J. M. (2010). Neighborhood structures for the container loading problem: A VNS implementation. *Journal of Heuristics, 16*(1), 1–22. https://doi.org/10.1007/s10732-008-9081-3.

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

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

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

Pollaris, H., Braekers, K., Caris, A., Janssens, G. K., & Limbourg, S. (2016). Capacitated vehicle routing problem with sequence-based pallet loading and axle weight constraints. *EURO Journal on Transportation and Logistics, 5*(2), 231–255. https://doi.org/10.1007/s13676-014-0064-2.

Post, S. (2020a). Swiss post expects a new all-time record in December. Accessed 30 July 2021 https://post-medien.ch/en/swiss-post-expects-a-new-all-time-record-in-december/.

Post, T. Z. (2020b). Post Office almost drowning in the flood of parcels. Accessed 30 July 2021 https://www.zug4you.ch/en/news/news-articles/a/post-office-almost-drowning-in-the-flood-of-parcels.

de Queiroz, T. A., & Miyazawa, F. K. (2013). Two-dimensional strip packing problem with load balancing, load bearing and multi-drop constraints. *International Journal of Production Economics, 145*(2), 511–530. https://doi.org/10.1016/j.ijpe.2013.04.032.

Research, & Markets (2018). Global logistics market 2017–2018 & 2023 - market is estimated to grow to $12.6 bn. Accessed 30 July 2021 https://www.prnewswire.com/news-releases/global-logistics-market-2017-2018-2023—market-is-estimated-to-grow-to-12-6-bn-300708730.html.

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

Statista (2020). Global parcel shipping volume between 2013 and 2026. Accessed 30 July 2021 https://www.statista.com/statistics/1139910/parcel-shipping-volume-worldwide/.

Terno, J., Scheithauer, G., Sommerweiß, U., & Riehme, J. (2000). An efficient approach for the multi-pallet loading problem. *European Journal of Operational Research, 123*(2), 372–381. https://doi.org/10.1016/S0377-2217(99)00263-5.

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

Trivella, A., & Pisinger, D. (2017). Bin-packing problems with load balancing and stability constraints. *INFORMS Transportation and Logistics Society Conference*.

Wäscher, G., Haußner, H., & Schumann, H. (2007). An improved typology of cutting and packing problems. *European Journal of Operational Research, 183*(3), 1109–1130. https://doi.org/10.1016/j.ejor.2005.12.047.

Zachariadis, E. E., Tarantilis, C. D., & Kiranoudis, C. T. (2009). A guided tabu search for the vehicle routing problem with two-dimensional loading constraints. *European Journal of Operational Research, 195*(3), 729–743. https://doi.org/10.1016/j.ejor.2007.05.058.

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