TOP (2021) 29:177–203
https://doi.org/10.1007/s11750-020-00577-8

ORIGINAL PAPER

Logistic constraints in container loading problems: the impact of complete shipment conditions

I. Gimenez-Palacios[1] · M. T. Alonso[1] · R. Alvarez-Valdes[2] · F. Parreño[1]

Received: 28 January 2020 / Accepted: 13 July 2020 / Published online: 24 July 2020

Abstract
Container loading problems (CLP), in which a set of boxes have to be loaded into containers or onto trucks, are at the core of many transportation problems. Good solutions to these packing problems are crucial for the efficient use of logistic resources. However, to produce solutions that are useful in practice, besides the basic conditions ensuring that boxes do not exceed the container dimensions and do not overlap each other, other physical and logistic constraints have to be included when solving the CLP. In this study, we focus on a special type of logistic constraint: complete shipment constraints, which ensure that either all or none of the boxes in each customer order are loaded. Although these constraints have seldom been considered in the literature, they arise very often in practice. Customers do not want to receive incomplete parts of their orders in successive deliveries and retailers do not want to have to store incomplete orders and wait until the remaining boxes arrive to deliver the orders to the customers. We include these constraints in an integer linear formulation of the CLP and develop four heuristic strategies to deal with them efficiently. An extensive computational study compares the relative performance of these strategies and shows that when a good strategy is used the impact of these constraints on the solution of the CLP is quite small.

Keywords Container Loading · Logistic Constraints · VNS · Integer models

✉ R. Alvarez-Valdes
ramon.alvarez@uv.es

I. Gimenez-Palacios
ivan.gimenez@uclm.es

M. T. Alonso
mariateresa.alonso@uclm.es

F. Parreño
francisco.parreno@uclm.es

[1] Department of Mathematics, University of Castilla-La Mancha, Albacete, Spain

[2] Department of Statistics and Operations Research, University of Valencia, Valencia, Spain

Mathematics Subject Classification 90C11 · 90C27 · 90C90

1 Introduction

Every day millions of products are sent to customers, using a wide range of means of transportation. These products are first packed in boxes and on pallets and then loaded onto trucks or into containers. In long-distance, transoceanic traffic, the use of containers has increased from 40 million TEUs (Twenty-foot Equivalent Units) in 1996 to 148 million TEUs in 2018 (UNCTAD 2018). Medium-range transportation is mostly carried out by trucks and trains. According to EUROSTAT, in 2017 road traffic between the 29 countries of the European Union amounted to 1.913.000 million tkm (tonne-kilometres), while trains moved 430.000 million tkm (EUROSTAT 2017). Most short-distance transportation is performed by small trucks, vans, and other non-polluting means of transportation in the case of city logistics.

Efficient use of these means of transportation requires solving a packing problem to decide how to place the small objects (boxes or pallets) in the large objects (containers or trucks) so to as maximize an objective, such as the total volume occupied or the total value of the objects packed. In the Cutting & Packing field, this problem is usually known as the Container Loading Problem (CLP). Therefore, in this study, without loss of generality, small objects will be called boxes and large objects containers. The basic version of the CLP includes only geometric constraints: boxes cannot exceed the dimensions of the container, and cannot overlap. These constraints define a challenging three-dimensional combinatorial problem which has been extensively studied in the scientific literature.

However, to be useful in practical applications, many other constraints have to be taken into account besides the basic geometric conditions. A list of these constraints was proposed by Bischoff and Ratcliff (1995). More recently, Bortfeldt and Wäscher (2013) revisited this list, proposing a classification of constraints and reviewing how they have been considered in the algorithms developed for the CLP. Practical constraints can be classified into two main groups, physical and logistic. Physical constraints are related to the physical characteristics of containers and boxes. They include weight, orientation, load-bearing, and stability constraints. Weight constraints limit not only the total weight, but also the weight on the axles in the case of trucks, as well as the load balance, forcing the weight to be evenly spread along the floor. Orientation constraints determine which rotations of the boxes are allowed and which are forbidden. Load-bearing constraints limit the weight each box can bear to avoid damage to the cargo. There are two types of stability constraints: static stability, when the container is not moving, and dynamic stability, when it is being moved and is subject to acceleration, braking and turns.

Logistic constraints concern the way in which boxes are delivered to customers. They include allocation, multi-drop, priority, and complete shipment constraints. Allocation constraints may take the form of connectivity constraints, forcing a set of boxes to be together in the same container, or separation constraints, forcing subsets of boxes to be in different containers. Multi-drop constraints arise when the container contains boxes for different customers to be unloaded in order along the route and force the boxes for a given customer to be accessed without moving boxes for clients to be served later. Priority constraints ensure that high-priority boxes are loaded before low-priority boxes if they do not all fit into the container. In a similar vein, complete shipment constraints ensure that for every order, either all or none of its boxes are loaded.

Bortfeldt and Wäscher (2013) found that some of the physical constraints, such as orientation, total weight limit, or static stability, had been used in many studies, while others, such as axle-load, load-bearing and dynamic stability, had seldom been considered. With regard to logistic constraints, multi-drop constraints had been considered in combined routing and packing problems, but the others had been almost entirely neglected. Bortfeldt and Wäscher (2013) also found that there were very few studies that included several constraints simultaneously, as they arise in practice. Since 2013, studies on the CLP have been considering more practical constraints, and there have been interesting advances, especially in relation to stability, axle-load and load-bearing constraints.

Nevertheless, some logistic constraints, such as complete shipment constraints, have received very little attention, although there are many situations in which these constraints are imposed. In the problem that inspired this study, a large furniture factory sends its products to retail shops which in turn serve their customers. Very often, customer orders include large items that are packed into several boxes to be loaded onto trucks. Customers and shops do not want to receive only a subset of these boxes and keep them in storage until the remaining boxes arrive. Similar situations arise when containers or trucks are shipped on a regular basis and it is always preferable to send complete orders rather than to fill the remaining space in containers or trucks with boxes that will have to wait undelivered until all the other boxes completing the order are shipped.

This paper focuses on these complete shipment constraints and their effect on the solutions of the CLP. We propose an integer formulation and develop four heuristic strategies to solve large instances. The computational study shows that when an efficient strategy is used the inclusion of these constraints produces only a small decrease in the volume occupied in the container.

The structure of the paper is as follows. The relevant literature on the CLP and practical constraints is reviewed in Sect. 2. Section 3 presents the definition of the problem and an integer formulation. A VNS algorithm for the basic CLP is described in Sect. 4 and the four heuristic strategies in Sect. 5. The computational study is summarized in Sect. 6 and conclusions and future work are discussed in Sect. 7

2 Related work

The single container loading problem (CLP) plays a central role in Cutting and Packing due to its many applications and extensions and it has been extensively studied. The basic geometric constraints, preventing the boxes from exceeding the dimensions of the container and from overlapping each other, already define a challenging combinatorial optimization problem which is NP-hard (Scheithauer 1992).

According to the typology proposed by Wäscher et al. (2007), it can be classified as a 3D-SLOPP (three-dimensional single large object placement problem), if the set of boxes is weakly heterogeneous, or a 3D-SKP (Single Knapsack Problem) if the set of boxes is strongly heterogeneous. Up to now, few exact solution approaches have been proposed (Fekete et al. 2007; Martello et al. 2000; Junqueira et al. 2012a). In contrast, many heuristic algorithms have been developed. Recent proposals can be classified into two groups: metaheuristics and tree-search algorithms. Metaheuristics include genetic algorithms (Gehring and Bortfeldt 2002), tabu search (Bortfeldt et al. 2003), simulated annealing (Mack et al. 2004), GRASP (Moura and Oliveira 2005; Parreño et al. 2008), and VNS (Parreño et al. 2010). Tree-search algorithms proposed in recent years have been shown to produce the best results for the CLP (Fanslau and Bortfeldt 2010; Zhu et al. 2012; Araya and Riff 2014; Araya et al. 2017).

Following the classification of practical constraints into physical and logistic constraints, we will now review the use of these constraints in recent years.

2.1 Physical constraints

A first group of physical constraints is related to weight. In almost all applications there is a limit to the weight that can be loaded into the container (Bortfeldt et al. 2003; Egeblad et al. 2010). In the case of products being transported by trucks with several axles, the weight that each axle can support is also limited (Lim et al. 2013; Pollaris et al. 2016; Alonso et al. 2019). The weight of the cargo has to be evenly distributed on the container floor. A simplified approach to this condition, commonly used, is to require that the centre of gravity of the cargo has to be as close as possible to the centre of the container (Bortfeldt and Gehring 2001). However, as Ramos et al. (2018) have shown, this constraint does not guarantee compliance with transportation regulations, so truck-specific load distribution diagrams have to be considered and the corresponding constraints included.

As for the boxes, the most common constraint is related to their orientation. In some cases, all six possible orientations of a box are allowed (Parreño et al. 2008), but in most practical situations just one vertical orientation is permitted (This side up!) and only 90° horizontal rotations are possible (Correcher et al. 2017; Toffolo et al. 2017). In some cases, rotation is not allowed at all (Junqueira et al. 2012b).

Load-bearing constraints limit the weight a box can support and are used to prevent boxes being damaged by excessive weight resting on them. They can be expressed in different ways: limiting the number of boxes that can be placed on top of others (Bischoff and Ratcliff 1995), classifying some boxes as fragile and prohibiting other boxes from being put above them (Paquay et al. 2016), or limiting the maximum weight a box can support per unit area (Alonso et al. 2014; Junqueira et al. 2012b).

Stability of the cargo is also a very important condition in practice. Static, or vertical, stability constraints ensure that loaded boxes do not fall when the container is not moving, especially during loading and unloading operations. Static stability has usually been considered by imposing full support conditions, in which the base of each box is completely supported by other boxes or by the container floor (Araujo and Armentano 2007; Fanslau and Bortfeldt 2010), or partial base support, in which the support can be reduced to a given percentage of the base (Jin et al. 2004; Junqueira et al. 2012b). More recently, Ramos et al. (2016) have shown that this approach may be too restrictive and have developed an alternative approach using mechanical equilibrium conditions. Dynamic, or horizontal, stability constraints are defined to ensure that boxes will not move, and therefore will not be damaged, when the container is moving and subjected to forces when accelerating, braking, or turning. Ramos et al. (2015) have proposed new metrics, extending the initial proposals of Bischoff and Ratcliff (1995). Alonso et al. (2017, 2019) have considered dynamic stability conditions in their models for multiple container loading.

2.2 Logistic constraints

Allocation constraints appear in many container loading problems. Sometimes, they are connectivity constraints, requiring a particular subset of boxes to be loaded into the same container (Liu et al. 2011), or relative positioning constraints, requiring certain subsets of boxes to be placed together in the container, to facilitate their delivery to specific customers (Makarem and Haraty 2010; Egeblad et al. 2010). In contrast, there are situations in which separation constraints are imposed, preventing incompatible types of products from being loaded into the same container (Eley 2003; Battarra et al. 2009).

Multi-drop conditions arise when the boxes to be loaded into the container belong to different customers that are visited in a given order. The items for each customer have to be placed together and they have to be accessed when the container reaches the customer without moving any box corresponding to other customers that will be visited later in the route. These constraints have been implemented in various ways. One way is by imposing the condition that the boxes must be visible, meaning that when the container reaches a customer, one side of each of their boxes must be completely visible from the entrance of the container (usually its back door) or blocked only by boxes for the same customer. This is the approach followed by Gendreau et al. (2006) in their tabu search algorithm for combined routing and packing problems, by Christensen and Rousøe (2009) in their tree-search algorithm, by Fuellerer et al. (2010) in their Ant Colony Optimization algorithm, by Ceschia and Schaerf (2013) in their local search procedure, and by Bortfeldt (2012), who developed a tabu search for the routing part and a tree search for the packing part. Liu et al. (2011) require not only visibility, but also reachability, considering the maximum distance at which a box can be accessed by the unloading devices. Junqueira et al. (2012b) define virtual walls separating boxes for different customers and include a parameter indicating up to what distance boxes for one customer can be located behind the wall separating them from the boxes for the next customer.

When the available space in the container is not enough to load all the boxes, a decision has to be made about which boxes are to be loaded and which left out. In some practical situations, boxes have priorities and loading high-priority boxes is preferred. Usually these priorities are based on established delivery dates, although they can also be related to the characteristics of the products (freshness, shelf-life). Ren et al. (2011) considered only two types, high and low priorities, and established that low-priority boxes should not be loaded if this leads to high-priority boxes being left behind. Wang et al. (2013) also consider high and low priorities and propose a beam search algorithm, imposing the condition that all high-priority boxes must be loaded. A particular type of priority is introduced by Jamrus and Chien (2016). In their problem, all the boxes fit into the container and priorities are used in their genetic algorithm to place the boxes with higher priority closer to the container door to be delivered earlier. Sheng et al. (2017) also consider two types of boxes, with and without expiry dates, and develop an iterative procedure to load as many orders composed of boxes with expiry dates as possible first, before considering loading orders of boxes without expiry dates.

In addition to priority restrictions, another situation that can arise when not all the boxes fit into the container is that boxes can be grouped into orders, and either all the boxes for the same order must be loaded or none at all. This is known as the complete shipment condition. It is usually considered in combined routing and packing problems, in which when building a route that includes several customers, all the boxes for each customer in the route have to fit into the container, also satisfying the multi-drop constraints. However, it has seldom been considered in pure container loading problems. It was listed by Bischoff and Ratcliff (1995) and mentioned as a possible extension by Eley (2003), but to the best of our knowledge it has only been implemented by Sheng et al. (2017). Their tree search packing procedure considers guillotine cuts, which is not the usual situation in container loading. In the light of the literature review, our objective is to study in depth how to incorporate complete shipment constraints efficiently into the container loading problem.

3 Loading with complete shipment constraints

In this problem, a container of dimensions (L, W, H) has to be filled with a set of boxes. There are n box types with dimensions $(l_j, w_j, h_j), j = 1, \dots, n$, and volume $v_j = l_j w_j h_j$. Boxes are grouped in m orders, corresponding to the clients' requirements. For each order $i = 1, \dots, m$, the number of required boxes of type j is n_{ij}. Therefore, the volume of order i is $V_i = \sum_j v_j n_{ij}$. The objective is to maximize the volume occupied by boxes in the container, subject to several types of constraints. In this study we are going to consider the basic geometric constraints, preventing boxes from exceeding the dimensions of the container and overlapping each other. In addition, the boxes have to be placed with their sides parallel to the sides of the container and each box has its own set of allowed orientations. We do not add any of the other constraints described above and focus on the effect of imposing the complete shipment constraints.

The problem can be formulated as an integer linear model, taking the base model by Junqueira et al. (2012a) (which is a direct extension of the model by Beasley (1985)) and adding constraints for complete shipment. A Cartesian coordinate system is used with the origin in the front-left-bottom corner of the container. In this system, (x, y, z) will be the coordinates of the front-left-bottom corner of a box. The possible positions along the axes are defined by sets $X = \{0, 1, \dots, L - \min_j(l_j)\}$, $Y = \{0, 1, \dots, W - \min_j(w_j)\}$, $Z = \{0, 1, \dots, H - \min_j(h_j)\}$, but these sets can be reduced to the positions of the normal patterns (Christofides and Whitlock 1977):

$$X = \{x \mid x = \sum_j \epsilon_j l_j, 0 \le x \le L - \min_j(l_j), \epsilon_j \in Z^+, j = 1, \dots, n\}, \tag{1}$$

$$Y = \{y \mid y = \sum_j \epsilon_j w_j, 0 \le y \le W - \min_j(w_j), \epsilon_j \in Z^+, j = 1, \dots, n\}, \tag{2}$$

$$Z = \{z \mid z = \sum_j \epsilon_j h_j, 0 \le z \le H - \min_j(h_j), \epsilon_j \in Z^+, j = 1, \dots, n\}. \tag{3}$$

In order to develop the non-overlapping constraints, we define the parameter:

$$c_{jxyzx'y'z'} = \begin{cases} 1, & \text{if a box of type } j \text{ placed with its front-left-bottom corner} \\ & \text{at } (x, y, z) \text{ occupies the position } (x', y', z') \\ 0, & \text{otherwise,} \end{cases}$$

which can be computed a priori for each (x, y, z) and (x', y', z') if the dimensions of each box type j are known.

The decision variables of the base model, a_{jxyz}, with $j = 1, \dots, n; x \in X, y \in Y, z \in Z$, are defined as

$$a_{jxyz} = \begin{cases} 1, & \text{if a box of type } j \text{ is placed with its front-left-bottom corner at} (x, y, z) \\ 0, & \text{otherwise.} \end{cases}$$

For the complete shipment constraints, we add variables:

$$r_{ij} = \text{the number of boxes of type } j \text{ assigned to order } i$$

$$z_i = \begin{cases} 1, & \text{if order } i \text{ is completely loaded into the container} \\ 0, & \text{otherwise.} \end{cases}$$

The model is:

$$\text{Maximize} \quad \sum_i V_i z_i, \tag{4}$$

$$st: \quad \sum_{j=1}^{n} \sum_{x\in X} \sum_{y\in Y} \sum_{z\in Z} c_{jxyzx'y'z'} a_{jxyz} \leq 1 \quad x' \in X, y' \in Y, z' \in Z, \tag{5}$$

$$\sum_{x\in X} \sum_{y\in Y} \sum_{z\in Z} a_{jxyz} = \sum_{i=1}^{m} r_{ij} \forall j, \tag{6}$$

$$r_{ij} \geq n_{ij} z_i \forall j, \forall i, \tag{7}$$

$$\begin{aligned} &a_{jxyz} \in \{0,1\}, j = 1, \dots, n; x \in X, y \in Y, z \in Z;, \\ &r_{ij} \in Z^+, i = 1, \dots, m, j = 1, \dots, n;\; z_i \in \{0,1\}, i = 1, \dots, m. \end{aligned} \tag{8}$$

The objective function (4) maximizes the total volume of the complete orders loaded into the container. Constraints (5) prevent the overlapping of boxes. Constraints (6) and (7) ensure the complete shipment conditions. Constraints (6) define the number of boxes of each type j assigned to each order i, and constraints (7) only allow an order to be considered complete if all its boxes are loaded.

This initial formulation could be enhanced in many ways. For instance, we could include a constraint limiting the total volume of the complete orders:

$$\sum_{i=1}^{m} V_i z_i \leq LWH \tag{9}$$

and cover inequalities could be derived from it, as in many other combinatorial optimization problems (Codas and Camponogara 2012; Shebalov et al. 2015; Dabia et al. 2019).

Another useful way of reducing the computational effort for solving the model is to reduce the possible positions of the boxes even further. Beyond the normal patterns used above, we could use raster points (Scheithauer and Terno 1996), reduced normal patterns (Boschetti and Mingozzi 2002), or meet-in-the-middle patterns (Côté and Iori 2018). A complete review and comparison of reduction methods can be found in de Almeida Cunha et al. (2019).

In the test instances proposed by Bischoff and Ratcliff (1995) and Davies and Bischoff (1999), the container has dimensions (587, 233, 220) and the box types range from 3 to 100. These instances have been used in most studies of the container loading problem and therefore they will also be used in our study. Considering that the model requires a variable a_{jxyz} for each possible position (x, y, z) and each box type j, even with the reductions mentioned above the number of variables is too high for the model to be solved in a reasonable time, even for small problems. In the following sections, several heuristic strategies will be described.

4 A VNS algorithm for container loading

In several of the procedures developed in the following sections, we use the Variable Neighborhood Search (*VNS*) algorithm developed by Parreño et al. (2010) as a starting point, with some extensions to make it more efficient. This algorithm has been shown to produce good results even when running for short times and it is flexible enough to be used and adapted in many ways as part of other algorithms developed to address the complete shipment constraints. The pseudocode of *VNS* appears in Algorithm 1.

The initial solution x is built by a constructive algorithm. It is an iterative procedure in which at each iteration two decisions are made. First, the maximal empty space nearest to a corner of the container is selected to be filled. Second, homogeneous blocks, that is, arrangements in rows and columns of the unpacked boxes of one box type, are considered for filling the selected space, and the block producing the largest increase in the occupied volume is chosen. This constructive algorithm is extremely fast and obtains quite good results that can be used as starting solutions for more elaborate procedures.

Six neighborhoods are used in the search:

- N_1: Block reduction
- N_2: Column insertion
- N_3: Box insertion
- N_4: Emptying a region and refilling it using the constructive algorithm
- N_5: Emptying a region and refilling it selecting at each step the block that best fits into the space
- N_6: Removing a percentage of the last boxes in the solution and refilling the space using the constructive algorithm

The moves can be classified into two groups. Moves defining N_2 and N_3 insert non-packed boxes into a solution, remove the boxes in the solution overlapping them, and fill the empty spaces using the constructive algorithm. The other four moves start by removing some elements of the solution, producing empty spaces that are merged with those existing in the solution, and then fill the empty spaces using either the constructive algorithm, as in N_1, N_4, and N_6, or another constructive algorithm with a different block selection criterion, as in N_5. Moves defining N_1, N_4, and N_6 differ in their impact on the solution. N_1 has a local impact on the region in which the reduced block was placed. The region to be emptied in N_4 (and also in N_5) is determined by selecting two empty spaces and building the minimum rectangle so as to include them. If these two empty spaces are widely separated, a large part of the container is emptied, making a global impact on the solution. In N_6 the impact depends on the percentage of blocks removed, but the way in which it removes them is completely different from N_4, thus producing a different type of move.

The six neighborhoods are used in the shaking phase, but only N_1 to N_5 in the local search. The stopping criterion is reaching a time limit, although the algorithm also stops if all the boxes are loaded into the container, as no improvement is then possible.

Algorithm 1 VNS algorithm

```
$N_p$= the set of neighborhood structures, for $p = 1, ..., p_{max}$ for the shaking phase
$N_l$= the set of neighborhood structures, for $l = 1, ..., l_{max}$ for the local search
$x$= Initial solution obtained by a constructive algorithm
$TimeLimit$ = Maximum running time
for ($Time < TimeLimit$) do
  $p = 1$
  while $p < p_{max}$ do
    (a) Shaking: Generate a neighbor $x'$ of $x$ at random, in $N_p$
    (b) Local Search:
    $l = 1$
    while $l < l_{max}$ do
      Find the best neighbor $x''$ of $x'$ in $N_l$
      if $x''$ is better than $x'$ then
        $x'$=$x''$
        $l = 1$
      else
        $l = l + 1$
      end if
    end while
    if $x'$ is better than $x$ then
      $x = x'$
      $p = 1$
    else
      $p = p + 1$
    end if
  end while
end for
```

Two extensions have been added to the original algorithm:

- *Score function* The Parreño et al. (2010) algorithm considered two criteria for selecting the block being packed at each step: the block that produces the largest increase in the volume occupied, and the block that fits best in the empty space, filling three, two, or one of its dimensions. More recently, Araya et al. (2017) have proposed a more complex score function which favours blocks that fit well into the container, considering the blocks previously placed and the empty spaces left. It also takes into account the volume of the block and an estimate of the expected waste that will be produced if the block is chosen, and tries to pack blocks composed of big boxes first. They have shown that this score function produces better results than previous selection criteria and we have therefore added it to our algorithm.
- *Block generation* The blocks considered in the original algorithm were exclusively homogeneous blocks: arrays of rows and columns using only one type of box, always with the same orientation. Later studies have shown that more general heterogeneous blocks, combining boxes of different types in different orientations, provide more flexibility to the search and thus better results (Fanslau and Bortfeldt 2010; Zhu et al. 2012). Consequently, we have extended the block generation process to consider heterogeneous blocks.

5 Solution approaches with complete shipment

This section presents several strategies developed for solving the container loading problem with complete shipment constraints heuristically.

5.1 A local a priori strategy: selecting the next order to load

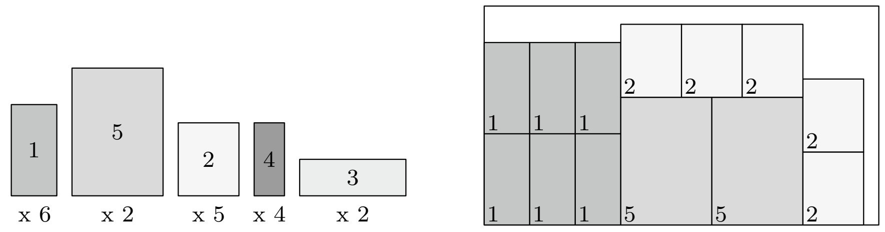
A simple loading strategy is to sort the orders into a list according to some criterion, such as volume or number of boxes, take the next order on the list, and try to load all its boxes using the constructive deterministic algorithm which is part of the VNS procedure. If all of them are loaded, the next order is considered. If all the boxes cannot be loaded, the boxes in the incomplete order that have been loaded are removed and the process moves on to the next order. When all the orders on the list have been considered, some of them will have been loaded, resulting in a feasible solution. The pseudocode of the procedure appears in Algorithm 2.

This can be seen as an a priori strategy, since at each step of the process the order being loaded has been decided beforehand. It is a fairly intuitive way of ensuring that complete orders are loaded, but as will be seen in the following sections, there are other alternative strategies. The procedure can be randomized if the list is constructed using a biased or unbiased random criterion. Initially, the boxes in an order will be placed close together in the container, which may be a desirable feature in the loading and unloading process, but later in the procedure the boxes will be loaded into the remaining empty spaces, which may be scattered throughout the container.

Algorithm 2 Local *a priori* strategy

```
$L$= a sorted list of orders
for ($i$=1 to $m$) do
    Take the next order $i$
    Call the constructive algorithm to load the boxes in $i$
    if (All boxes in $i$ are loaded) then
        Update the partial solution
    else
        Remove the boxes of order $i$
    end if
end for
```

In Fig. 1a and b, we can see an example of this procedure. We have to load five orders, sorted by non-increasing volume in Fig. 1a. Each order is composed of only one box type and the number of boxes in the order appears below each box. Figure 1b shows the final result, in which orders 1, 5, and 2, in this order, are fully loaded into the truck, but orders 4 and 3 cannot be fitted in it. Some of the boxes in order 4 could be loaded, but not the complete order of four boxes.



(a) Orders sorted by non-increasing volume

(b) Loading complete orders according to the sorted list

Fig. 1 An example of the local a priori strategy

5.2 A global a priori strategy: selecting a set of orders to load

Instead of selecting one order at a time to be loaded, an alternative consists in selecting a subset of orders to be loaded together. Let TV_1 be the total volume occupied in the solution obtained by running the *VNS* algorithm without the complete shipment constraints. This value TV_1 is an upper bound on the total volume occupied in a solution in which complete shipment is enforced if the *VNS* is used for loading. Therefore, a set of orders that could potentially be completely loaded into the container can be obtained by solving a knapsack problem:

$$\text{Maximize} \quad \sum_i V_i s_i, \tag{10}$$

$$st: \quad \sum_i V_i s_i \leq TV_1, \tag{11}$$

$$s_i \in \{0, 1\} \forall i. \tag{12}$$

Let S_1 be the set of orders in the solution of this initial knapsack problem. We then run the *VNS* algorithm again to fill the container, but only with the boxes belonging to the S_1 orders. There will be two possible outcomes: if all these boxes fit into the container, the process ends with this feasible solution; otherwise, the container will be filled to a total volume of TV_2, and some of the S_1 orders will be incomplete. To select a new set of orders to be considered for loading, we solve the integer model (10)–(12) again, replacing the right-hand side of (11) by TV_2. The solution will be a new set S_2 and the *VNS* algorithm runs with the boxes of the orders in S_2. Again there will be two possible results. If all the boxes do not fit into the container, the integer model is run again with the right-hand side given by the total occupied volume in the last application of *VNS*. This process is repeated as many times as necessary until all the boxes in a set S_k fit into the container. The first time the boxes in a set S_k fit into the container, instead of stopping, we try to recover part of the last decrease and solve the integer model again with a right-hand side of $(TV_{k-1} + TV_k)/2$ in constraint (11). If the boxes in the set of orders obtained in the solution fit, the *RHS* is increased again to the midpoint between TV_k and $(TV_{k-1} + TV_k)/2$. The first time the boxes in the selected orders do not fit, the process ends with the last feasible solution obtained.

If the allowed runtime is *TimeLimit*, we set a maximum number of iterations *MaxIter* and give a maximum time of *TimeLimit*/*MaxIter* to the first application of the *VNS* and subsequent iterations in which the *RHS* has been decreased to allow the total volume to decrease as much as necessary to find a feasible solution with a set of complete orders in the container. The first time we obtain a set of orders whose boxes fit into the container and the *RHS* is increased, we give the *VNS* all the remaining time. If the *VNS* finds a new feasible solution without running out of time, the *RHS* is increased again, as explained above, until the time is exhausted.

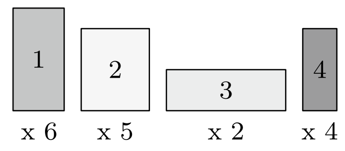
Algorithm 3 Global a priori strategy

```
$TV_1$= volume occupied by the solution of $VNS$ without complete shipment constraints
$V_{sol} = 0$, volume of the best solution obtained
$Model$ = integer linear model defined by expressions (10)-(12)
$MaxIter$, $TimeLimit$
$k = 1$
$fit = false$
while ($fit = false$ ) do
    Determine a set of orders $S_k$ solving $Model$ with $VT_k$ as the $RHS$ in constraint (11)
    Call $VNS$ to load the boxes of the orders in $S_k$ with time limit $TimeLimit/MaxIter$
    if (Not all boxes fit into the container) then
        $k = k + 1$
        $TV_k$= volume occupied by the solution of $VNS$
    else
        $fit = true$
        if ($k = 1$) then
            Return $V_{sol} = \sum_{i \in S_k} \sum_j v_i n_{ij}$
        end if
    end if
end while
while ($fit = true$ ) do
    $k = k + 1$
    Determine a set of orders $S_k$ solving $Model$ with the midpoint of the last two $TV$ used
    as the $RHS$ in constraint (11)
    Call $VNS$ to load the boxes of the orders in $S_k$ with all the remaining running time
    if (Not all boxes fit into the container) then
        Return the best feasible solution obtained, $V_{sol} = \sum_{i \in S_{k-1}} \sum_j v_i n_{ij}$
    end if
end while
```

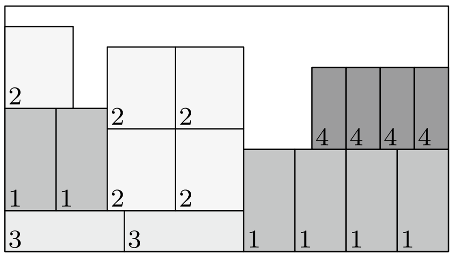
An example is shown in Fig. 2a and b, using the same data for five orders already used in Fig. 1a. The integer model (10)–(12) selects the orders in Fig. 2a to be loaded. In Fig. 2b, all these orders have been loaded into the container by the VNS algorithm.

5.3 An a posteriori strategy

A different approach will be first to load as many boxes as possible into the container and then to assign the loaded boxes to orders in an a posteriori strategy. As orders



(a) Orders selected by the integer model



(b) Loading the complete orders selected

Fig. 2 An example of the global a priori strategy

are usually composed of boxes of the same types, from a solution with a high volume occupancy and many boxes it could be possible to obtain a large set of complete orders, at least for homogeneous instances with few box types. For the first step, we can use the *VNS* algorithm described in Sect. 4 with an appropriate time limit. This algorithm produces solutions with high volume utilization percentages. Let n_j be the number of boxes of type j in the solution of the *VNS*. The optimal assignment of boxes to orders can be obtained by solving an integer linear model. If we define variables

$$z_i = \begin{cases} 1, & \text{if order } i \text{ is completely loaded} \\ 0, & \text{otherwise} \end{cases}$$

$$x_{ij} = \text{number of type } j \text{ boxes assigned to order} i,$$

the model is:

$$\text{Maximize} \quad \sum_i V_i z_i, \tag{13}$$

$$st: \quad \sum_i x_{ij} = n_j \forall j, \tag{14}$$

$$x_{ij} \geq z_i n_{ij} \forall i, \forall j. \tag{15}$$

The objective function (13) maximizes the volume of complete orders in the container. Constraints (14) ensure that only the boxes in the solution are assigned to orders. Constraints (15) allow an order i to be considered complete ($z_i = 1$) only if all n_{ij} boxes of each type j forming the order i are assigned to it. Although all the boxes in the solution are assigned by constraint (14), those belonging to incomplete orders are considered unassigned, so after identifying the complete orders in the solution there will be u_j unassigned boxes of type j. In fact, they will have to be removed from the container to get a feasible solution composed only of complete orders.

This a posteriori assignment strategy does not guarantee a large number of complete orders by itself. Boxes have been selected in the VNS algorithm without considering to which order they belong and it can be difficult to assign them to form complete orders, especially in strongly heterogeneous problems.

In order to improve the solutions obtained by this strategy, we have developed a procedure that divides the set of incomplete orders in the solution into two sets: S^C, the set of orders to be completed, and S^R, the set of orders whose boxes in the solution will be removed to make room for the boxes of orders in S^C. To determine S^C, we solve an integer problem to assign the unassigned boxes u_j, removing the orders that have been completed:

$$\text{Minimize} \quad \sum_i V_i w_i, \tag{16}$$

$$st: \quad \sum_i x_{ij} = u_j \forall j, \tag{17}$$

$$x_{ij} \leq w_i n_{ij} \forall i, \forall j. \tag{18}$$

Constraints (17) force all the unassigned boxes in the solution to be assigned to orders. In constraints (18), variable w_i takes value 1 as soon as one box of any type j is assigned to order i. The objective function minimizes the volume of orders with $w_i = 1$, so in the solution we expect to find the boxes mostly assigned to almost complete orders, although orders with just a few boxes cannot be ruled out because all unassigned boxes must be assigned.

The construction of set S^C follows an iterative process in which initially $S^C = \emptyset$ and all orders i with $w_i = 1$ are included in S^R. At each iteration, one of these three criteria is randomly chosen:

- C1: The volume of the boxes that should be loaded into the container to complete the order: $R_i = \sum_j v_j \max\{n_{ij} - u_j, 0\}$,
- C2: The proportion of this volume with respect to the total volume of the order, $P_i = R_i / V_i$,
- C3: The volume of the unassigned boxes that can be assigned to the order, $A_i = \sum_j v_j a_{ij}$, where:

$$a_{ij} = \begin{cases} n_{ij}, & \text{if } n_{ij} < u_j \\ u_j, & \text{if } n_{ij} \geq u_j. \end{cases}$$

The best order in S^R according to the chosen criterion is removed from it and included in S^C if the volume of the boxes to be loaded to complete the orders in S^C, multiplied by a factor randomly chosen in the interval [1.5, 3], is lower than the volume of the boxes belonging to the orders in S^R that will be removed. We consider the volume of the boxes multiplied by a factor larger than 1 because we have to take into account that the empty spaces created in the container by removing boxes can sometimes be difficult to use for loading boxes, especially in the case of strongly heterogeneous boxes with many different box dimensions.

Once set S^C has been determined, all the boxes belonging to orders still in S^R are removed. If there are several boxes of one type and only some of them have to be removed, those in the topmost positions are selected, trying to leave the empty spaces as high as possible in the container. The boxes required to complete the orders in S^C are then loaded using a randomized constructive algorithm (Parreño et al. 2008). If not all of them can be loaded, the orders in S^C are considered for loading one at a time, in a random sequence. The pseudocode of the procedure appears in Algorithm 4.

Algorithm 4 An *a posteriori* strategy

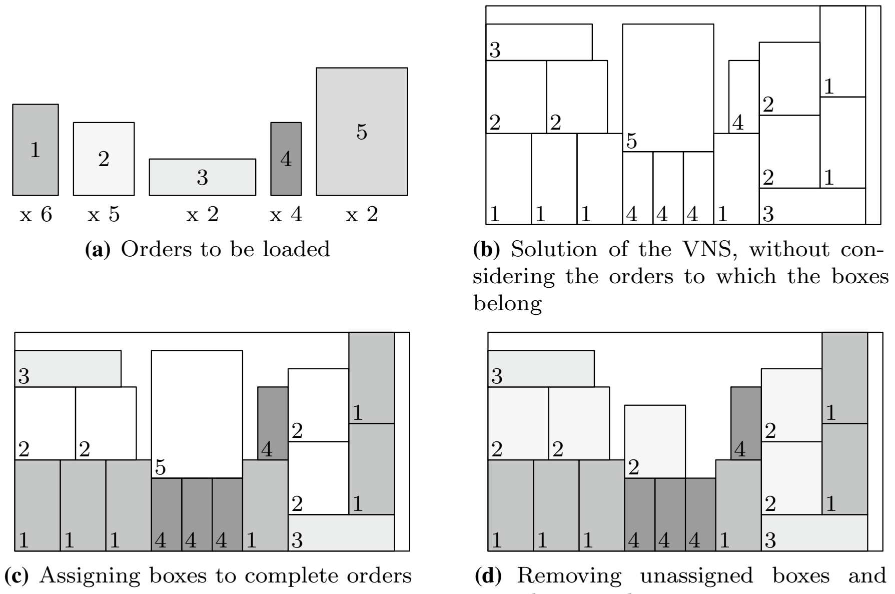
```
$Model_1$ = integer linear model defined by expressions (13)-(15)
$Model_2$ = integer linear model defined by expressions (16)-(18)
PHASE 1 {Fill the container and identify complete orders}
Solve the instance with the $VNS$ without complete shipment constraints
Use $Model_1$ to assign the loaded boxes to orders, producing:
  $S_1$ = set of complete orders
  $S_2$ = set of incomplete orders
  $S_3$ = set of orders without any box assigned
PHASE 2 {Determine the set of orders to complete}
if ($S_2 \cup S_3 \neq \emptyset$) then
  Solve $Model_2$ over the orders in $S_2 \cup S_3$
  $S^R = \{i \mid w_i = 1$ in the solution of $Model_2\}$, $S^C = \emptyset$
  $V^R$ = volume of the boxes of orders in $S^R$ loaded in the container
  $V^C$ = volume of the boxes if orders in $S^C$ not loaded in the container
  Take $\lambda$ at random in [1.5, 3]
  while ($S^R \neq \emptyset$) do
    if ($\lambda V^C < V^R$) then
      Select one criterion at random among C1, C2, C3
      Select $k$, the best order in $S^R$ according to the chosen criterion
      Add $k$ to $S^C$, remove it from $S^R$ and update $V^C$ and $V^R$
    else
      Go to Phase 3
    end if
  end while
else
  Return $S_1$
end if
PHASE 3 {Complete the orders}
Remove the boxes of orders in $S^R$ from the container
Load the remaining boxes of orders in $S^C$ using a randomized constructive algorithm
if (Not all boxes can be loaded) then
  Take orders in $S^C$ one at a time, at random, and try to load its remaining boxes
end if
Return $S_1$ and the orders in $S^C$ that have been completed
```

Figure 3a–d shows an example of this strategy. The five orders in Fig. 3a have to be loaded. Figure 3b shows the solution obtained by the VNS algorithm. In Fig. 3c, some of the boxes have been assigned by solving model (13)(15), completing orders 1, 3, and 4. White boxes are unassigned. In the second step, the three criteria will produce $S^C = \{2\}$ and $S^R = \{5\}$. Removing the box belonging to order 5, the empty space can be used to load the remaining box in order 2.

5.4 A mixed strategy

An alternative to the previous strategy is not to fill the container up completely, but only to a given percentage of its volume. When this percentage is reached, the filling process stops and the model of expressions (13)–(15) is solved to assign the loaded boxes to orders. Orders i in the solution that are not complete ($z_i = 0$), but have some boxes in the solution ($\exists j \mid x_{ij} \geq 1$), are considered together for completion, using a deterministic constructive algorithm. If all these boxes are loaded, the remaining orders are considered for loading, one at a time, selected using the three criteria described in the previous strategy. If not all the boxes can be loaded, all orders, with or without boxes in the partial solution, are considered for loading, one at a time, using the same procedure.

There are two main differences from the previous strategy. On the one hand, as the filling process does not go up to the whole volume of the container, the *VNS* algorithm with its improving procedures cannot be applied. Instead, we use a randomized constructive algorithm. On the other hand, as there will still be more empty space, it should be easier to complete orders and even to consider new orders that did not have any box in the partial solution. The pseudocode of the procedure appears in Algorithm 5.



(a) Orders to be loaded

(b) Solution of the VNS, without considering the orders to which the boxes belong

(c) Assigning boxes to complete orders

(d) Removing unassigned boxes and completing orders

Fig. 3 An example of the a posteriori strategy

Algorithm 5 A mixed strategy

```
$Model$ = integer linear model defined by expressions (13)-(15)
$\rho$ = percentage of the volume filled in the first phase
$TimeLimit$
while ($time < TimeLimit$) do
    PHASE 1 {Partial filling of the container}
    Apply a randomized constructive algorithm without complete shipment constraints
    until the percentage of volume filled reaches $\rho$
    Use $Model$ to assign the loaded boxes to orders, producing:
        $S_1$ = set of complete orders
        $S_2$ = set of incomplete orders
        $S_3$ = set of orders without any box assigned
    PHASE 2 {Completing the load}
    if ($S_2 \cup S_3 \neq \emptyset$) then
        Apply a constructive algorithm to load the remaining boxes of orders in $S_2$
        if (All boxes are loaded) then
            while ($S_3 \neq \emptyset$) do
                Select one criterion at random among C1, C2, C3
                Take $k$, the best order in $S_3$ according to the chosen criterion
                Load the boxes of order $k$ using a constructive algorithm
            end while
        else
            while ($S_2 \cup S_3 \neq \emptyset$) do
                Select one criterion at random among C1, C2, C3
                Take $k$, the best order in $S_2 \cup S_3$ according to the chosen criterion
                Load the remaining boxes of order $k$ using a constructive algorithm
            end while
        end if
    end if
end while
```

In Fig. 4a–c, we can see an example of the mixed strategy. In Fig. 4a, the truck is filled up to 50% with the VNS algorithm (phase 1). In Fig. 4b, model (13)–(15) assigns boxes to orders 1 and 3, and order 2 is incomplete. In Fig. 4c, order 2 is completed, and finally the complete order 5 is also introduced with the deterministic constructive algorithm (phase 2).

6 Computational study

We carried out an extensive computational analysis to study the effect of complete shipment conditions on the loading plans and to compare the performance of the strategies developed here to implement them.

We focused on answering the following research questions:

1. What is the effect of including complete shipment constraints in container loading? Is the volume occupied severely reduced when these constraints are added?
2. Which of the strategies developed to consider these constraints obtains the best results?
3. How are the solutions affected by the size and heterogeneity of the orders?

Our algorithm was implemented in C++11 on Visual Studio 2017 to Linux and compiled with g++. The computer used was a Linux OS machine, Linux version 3.10.0-693.5.2.el7.x86-64, gcc version 4.8.5 20150623, Red Hat 4.8.5-16 (GCC), with 1 Core at 2.40GHz, 4GB of RAM. For solving the integer linear programs we used CPLEX 12.8.0.0.

6.1 Test instances

The standard benchmark for the container loading problem was initially proposed by Bischoff and Ratcliff (1995) and then completed by Davies and Bischoff (1999). There are 1500 instances, classified into 15 classes, BR1 to BR15, with 100 instances per class. The numbers of box types per class are 3, 5, 8, 10, 12, 15, 20, 30, 40, 50, 60, 70, 80, 90, 100, so the instances are weakly heterogeneous in the first classes and evolve to strongly heterogeneous in the last classes. Usually, BR1–BR7 are considered weakly heterogenous and BR8–BR15 strongly heterogeneous. There is one container of dimensions (587, 233, 220) cm. The boxes generated in each instance completely fill the volume of the container, without considering three-dimensional geometric constraints, with an average of around 130 boxes per instance.

Instances BR1–BR15 have been adapted to the complete shipment case by grouping the boxes into orders. If m is the number of orders in the instance, $m \in \{10, 15, 20, 25, 50\}$, the order to which each box belongs is chosen at random in the interval $[1, m]$, guaranteeing that each order has at least one box. The averages of the numbers of boxes per order for the different numbers of orders appear in Table 1. The lower the number of orders, the larger the number of boxes in each order, and this could make it difficult to load complete orders.

Running times are adapted to the heterogeneity of the class. For each instance of class k the running time of the algorithms applied to it will be $300 + 60k$ CPU seconds. On the homogeneous instances in Class 1, with only 3 box types, the algorithms will run for 360 s, while longer times are assigned to strongly heterogeneous classes, acknowledging their special difficulty. On Class 15, with 100 different box types, the algorithms will run for 1200 s. Running times are rather long because this study has been designed to assess the effect of the complete shipment constraints and not to obtain solutions in short running times. By design, the proposed algorithms dealing with the complete shipment constraints will run up to the time limit. The constructive algorithm, which is used in different ways by those algorithms, is extremely fast, while the VNS algorithm, also used in several ways in the proposed algorithms, is able to obtain good results even when it is only allowed to run for very short times.

6.2 Performance of the *VNS* algorithm

Table 2 shows the performance of the *VNS* algorithm for each class of instances and for different time limits, without complete shipment constraints. It can be seen

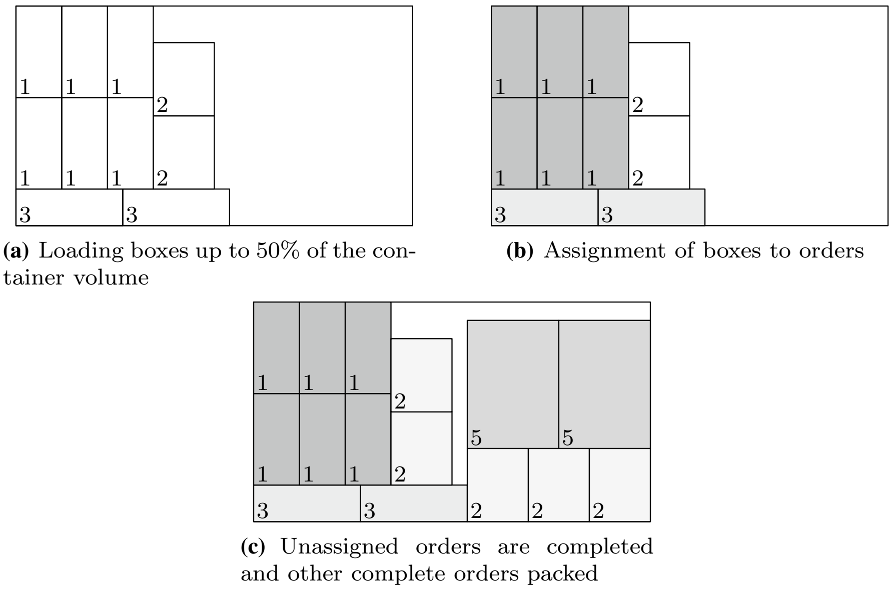

1
1
1
2
1
1
1
2
3
3
(a) Loading boxes up to 50% of the container volume
1
1
1
2
1
1
1
2
3
3
(b) Assignment of boxes to orders
1
1
1
2
5
5
1
1
1
2
3
3
2
2
2
(c) Unassigned orders are completed and other complete orders packed

Fig. 4 An example of the mixed strategy

that a significant improvement compared with the constructive algorithm is obtained by running *VNS* for 100 seconds. Nevertheless, increasing the running times only produces small improvements, as is usually the case in procedures based on local search. In each column, for a fixed number of orders, the volume decreases as heterogeneity increases, with differences of around 3% between classes BR1 and BR15.

6.3 Comparing the strategies

6.3.1 Results of the local a priori strategy

Table 3 contains the results of the local a priori strategy in which orders are taken from a list and considered for loading one at a time. Although we considered deterministic criteria for building lists, such as total volume or number of boxes, they did not obtain better results than the repeated execution of purely random lists, and the results in the table were obtained by considering random lists until the allowed running time was exhausted. The results exhibit three main characteristics. First, the number of orders does not produce any difference, as can be observed by looking at each row. Second, the percentage of volume decreases with increasing heterogeneity, as the columns show. Third, and most important, containers are only filled up to 80%, which is on average more than 13% less than the volume occupied when the VNS algorithm is used without considering the orders to which boxes belong. It is clear that this strategy, using a simple constructive algorithm and having to select among the boxes for only one order at each step, is not adequate when complete shipment constraints are included in the container loading problem.

6.3.2 Results of the global a priori strategy

The results of the global a priori strategy, in which at each iteration a subset of orders is selected to be loaded by solving a knapsack problem, appear in Table 4. Unlike the previous Table 3, here the results improve slightly with the number of orders, confirming that orders with many boxes are more difficult to complete that orders composed of a few boxes. The other characteristic, that the volume occupied in the container decreases on average as the heterogeneity increases, is shared with all the other strategies. What is especially important in Table 4 is that the average volume occupancy is extremely high, with values of almost 90% for the most difficult instances with few orders and strong heterogeneity, and values exceeding 95% for the easiest instances, homogeneous and with many orders composed of few boxes.

If these results are compared with those in the last column in Table 2, showing the results of the *VNS* algorithm running for a long time and without the complete shipment constraints, it can be observed that the percentages of volume occupancy decrease by less than 3% for the most difficult instances and the difference falls to 2% when the number of orders increases. The results of this strategy clearly show that including the complete shipment conditions in the standard container loading problem does not have to result in a large decrease in container occupancy if a good strategy is used.

6.3.3 Results of the a posteriori strategy

Table 5 shows the results of the a posteriori strategy. There are two columns for each number of orders and for all instances together. *Ph.*1 columns show the results of the initial strategy, using the VNS algorithm without considering the orders to which the boxes belong and then assigning boxes to orders, maximizing the volume of orders completed. It can be observed that this strategy works well for the first classes, which correspond to homogeneous instances. As expected, in solutions with many boxes of few types, it is possible to complete many orders because they are basically composed of the same types of boxes. In contrast, the volume percentages of complete orders decrease sharply in progressively more heterogeneous instances, especially when the number of orders is small

Table 1 Average number of boxes per order for each number of orders

Boxes per order	10 orders	15 orders	20 orders	25 orders	50 orders
Minimum	10.2	4.9	2.0	1.7	1.0
Average	13.3	8.8	6.6	5.3	2.7
Maximum	17.4	14.9	12.7	10.9	6.9

Table 2 Performance of the VNS for different time limits

		VNS			
Class	Constructive	100 s	200 s	500 s	1000 s
1	86.30	95.51	95.55	95.55	95.62
2	90.85	96.31	96.45	96.51	96.54
3	89.22	96.08	96.24	96.48	96.51
4	89.75	95.86	95.96	96.38	96.50
5	90.33	95.63	95.78	96.02	96.07
6	89.50	95.32	95.43	95.65	95.76
7	90.14	94.65	94.76	94.98	95.19
8	90.25	93.73	94.03	94.33	94.43
9	90.33	93.23	93.29	93.72	94.02
10	90.04	92.59	92.80	93.35	93.49
11	89.60	92.58	92.93	93.27	93.67
12	89.37	91.98	92.64	93.04	93.21
13	89.41	91.83	92.32	92.44	92.69
14	88.89	91.93	92.18	92.68	92.76
15	89.65	91.81	92.15	92.73	92.88
Average	89.56	93.94	94.17	94.48	94.62
Standard deviation	2.46	1.83	1.73	1.61	1.56

Table 3 Performance of the local a priori strategy

Class	10 orders	15 orders	20 orders	25 orders	50 orders	Overall	Std Dev
1	85.8	84.7	84.0	83.2	81.8	83.9	2.3
2	83.0	82.5	82.0	81.5	81.0	82.0	1.4
3	81.7	81.2	81.2	81.0	80.8	81.2	1.1
4	81.1	80.8	80.9	80.7	80.8	80.9	1.1
5	80.8	80.6	80.7	80.7	80.7	80.7	1.1
6	80.3	80.4	80.6	80.5	80.7	80.5	1.0
7	80.2	80.2	80.3	80.4	80.5	80.3	0.9
8	80.0	80.1	80.2	80.3	80.6	80.2	0.8
9	79.9	80.0	80.2	80.2	80.6	80.2	0.7
10	79.9	80.0	80.2	80.3	80.5	80.2	0.8
11	79.9	79.9	80.1	80.2	80.5	80.1	0.7
12	79.9	80.0	80.2	80.3	80.5	80.2	0.7
13	80.0	80.0	80.1	80.3	80.6	80.2	0.7
14	79.8	80.0	80.1	80.2	80.5	80.1	0.7
15	79.9	80.1	80.2	80.3	80.6	80.2	0.7
Average	80.8	80.7	80.8	80.7	80.7	80.7	
Std Dev	1.9	1.6	1.4	1.2	1.0		

and therefore each order is composed of many boxes. *Ph*.2 columns show that the improvement phase, in which some of the incomplete orders are chosen for completion and the others for removal, obtains much better solutions, most significantly for strongly heterogeneous instances with orders including many boxes, with an overall improvement of 6% in total volume occupancy. Nevertheless, even with the improvement phase, the results of this strategy are clearly worse l those of the previous global a priori strategy.

6.3.4 Results of the mixed strategy

Table 6 presents the results of the mixed strategy. In order to test whether the percentage to which the container is filled in the initial phase of the procedure influenced the results, several percentages (50%, 60%, 70%) were tested, but they did not offer significantly different results; therefore the table only shows the results for the $\rho = 50\%$ filling percentage. Unlike other procedures, there are almost no differences with respect to the number of orders, but as in other cases, the volume occupancy decreases with the heterogeneity of the instances. However, the results are not as good as those obtained in previous cases. The main reason may lie in the fact that the structure of the proposed procedure does not allow the use of the powerful VNS algorithm, replaced here by a randomized constructive algorithm. The flexibility obtained by not completely filling the container, leaving enough empty space for completing orders, does not seem to offset the initial disadvantage.

Table 4 Performance of the global a priori strategy

Class	10 orders	15 orders	20 orders	25 orders	50 orders	Overall	Std Dev
1	92.5	94.6	95.2	95.4	95.6	94.7	1.9
2	92.7	95.1	95.5	95.7	95.9	95.0	1.6
3	92.7	95.1	95.4	95.5	95.8	94.9	1.4
4	92.8	94.9	95.1	95.1	95.6	94.7	1.3
5	92.8	94.7	94.7	94.8	95.2	94.4	1.1
6	92.7	94.3	94.2	94.3	94.7	94.0	1.0
7	92.6	93.5	93.5	93.6	93.9	93.4	0.8
8	91.9	92.2	92.1	92.1	92.6	92.2	0.8
9	91.5	91.5	91.5	91.7	91.8	91.6	0.8
10	91.2	91.2	91.1	91.0	91.4	91.2	0.8
11	90.8	90.7	90.6	90.6	90.9	90.7	0.8
12	90.7	90.5	90.4	90.4	90.9	90.6	0.8
13	90.4	90.2	90.2	90.3	90.5	90.3	0.7
14	90.2	90.1	90.0	90.2	90.3	90.2	0.8
15	90.0	89.9	89.9	89.7	90.0	89.9	0.8
Average	91.7	92.6	92.6	92.7	93.0	92.5	
Std Dev	1.3	2.2	2.3	2.4	2.4		

Table 5 Performance of the a posteriori strategy and its improvement phase

	10 orders		15 orders		20 orders		25 orders		50 orders		Overall		Std Dev	
Cl.	Ph.1	Ph.2	Ph.1	Ph.2	Ph.1	Ph.2	Ph.1	Ph.2	Ph.1	Ph.2	Ph.1	Ph.2	Ph.1	Ph.2
1	92.0	92.0	94.0	94.0	94.9	94.9	95.2	95.2	95.7	95.7	94.3	94.3	2.1	2.1
2	91.6	91.7	93.8	93.8	94.7	94.7	95.2	95.2	96.1	96.1	94.3	94.3	2.1	2.1
3	90.6	90.7	92.7	92.7	94.0	94.1	94.7	94.7	96.1	96.1	93.6	93.7	2.4	2.4
4	89.4	89.8	91.7	91.9	92.9	93.2	94.1	94.2	95.7	95.7	92.8	93.0	3.0	2.7
5	87.9	88.8	90.7	91.2	92.4	92.6	93.0	93.3	95.2	95.2	91.8	92.2	3.6	2.9
6	85.4	86.7	89.1	89.9	90.7	91.5	91.9	92.2	94.7	94.8	90.4	91.0	4.1	3.5
7	81.5	84.0	86.3	87.7	88.1	89.3	89.9	90.6	93.4	93.5	87.8	89.0	5.0	4.0
8	75.1	82.8	80.4	85.0	82.9	86.4	85.3	88.2	90.9	91.3	82.9	86.7	6.7	3.9
9	69.5	82.6	75.5	84.3	80.0	85.7	82.9	86.6	89.1	89.9	79.4	85.8	8.2	3.6
10	65.2	81.4	72.4	83.9	76.3	84.6	79.4	86.1	87.8	89.1	76.2	85.0	8.9	3.8
11	63.0	81.8	71.0	83.5	74.2	84.4	78.1	85.5	86.4	88.2	74.5	84.7	9.4	3.4
12	57.0	81.4	66.8	83.8	71.3	84.4	75.7	84.8	84.8	87.2	71.1	84.3	10.8	3.2
13	54.7	81.0	64.1	83.4	68.6	83.7	73.2	84.9	83.8	86.9	68.9	84.0	11.2	3.2
14	54.7	81.3	63.5	83.1	68.4	84.1	73.3	84.7	83.9	86.6	68.8	84.0	11.7	3.0
15	52.4	80.6	61.8	83.0	66.6	83.5	72.5	84.3	83.3	86.1	67.3	83.5	12.0	3.1
Av.	74.0	85.1	79.6	87.4	82.4	88.5	85.0	89.4	90.5	91.5	82.3	88.4		
SD	15.8	5.2	12.8	4.9	11.2	4.9	9.4	4.6	5.4	4.1				

Table 6 Performance of the mixed strategy

Class	10 orders	15 orders	20 orders	25 orders	50 orders	Overall	Std Dev
1	90.4	90.9	90.8	90.9	90.1	90.6	2.1
2	89.7	89.9	90.1	90.3	89.9	90.0	1.4
3	88.3	88.9	89.3	89.4	88.9	89.0	1.2
4	87.6	88.2	88.7	88.8	88.3	88.3	1.3
5	86.6	87.7	88.3	88.5	87.8	87.8	1.3
6	85.7	87.2	87.6	87.9	87.2	87.1	1.3
7	84.7	86.4	87.0	87.1	86.8	86.4	1.3
8	84.6	86.3	86.8	87.2	87.0	86.4	1.3
9	84.1	85.7	86.5	86.8	86.7	85.9	1.4
10	83.8	85.3	86.1	86.4	86.6	85.6	1.4
11	83.6	84.9	85.7	86.3	86.4	85.4	1.4
12	83.3	84.6	85.5	86.1	86.4	85.2	1.5
13	83.1	84.3	85.1	85.8	86.4	84.9	1.5
14	82.7	84.0	85.1	85.7	86.3	84.7	1.6
15	82.2	83.9	84.8	85.5	86.3	84.6	1.7
Average	85.4	86.5	87.2	87.5	87.4	86.8	
Std Dev	2.9	2.4	2.1	1.9	1.6		

7 Conclusions

Complete shipment constraints, ensuring that either all the boxes in a customer order or none at all are packed in the container, have to be explicitly included in container loading problems to produce useful solutions in many practical situations. As the integer linear model including these constraints may be too large to be solved for real world instances, we have explored several different strategies to solve the problem heuristically. In line with the findings of Sheng et al. (2017), our extensive computational study shows that strategies based on selecting a subset of orders by using an iteratively modified knapsack problem are the most efficient. The other main finding of the study is that imposing complete shipment constraints does not produce large decreases in the volume occupied in the container, if an appropriate strategy is used. The code and the new instances generated can be accessed at https://github.com/ivangipa/CLPCS.

This study could be extended in a number of ways. Our experience and previous work on container loading problems have basically involved adding practical constraints progressively to the basic problem. In this line of study, our future work will add these complete shipment constraints to other practical constraints, such as load balance and static stability, and see how they interact and how the strategies developed here can be applied or adapted to more complex situations.

Acknowledgements This work has been partially supported by the Spanish Ministry of Science, Innovation, and Universities, project RTI2018-094940-B-I00, partially financed with FEDER funds and by the Junta de Comunidades de Castilla-La Mancha, project SBPLY/17/180501/000282, partially financed with FEDER funds.

Compliance with ethical standards

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

References

Alonso MT, Alvarez-Valdes R, Parreño F, Tamarit JM (2014) A reactive GRASP algorithm for the container loading problem with load-bearing constraints. Eur J Ind Eng 8(1):669–694

Alonso MT, Alvarez-Valdes R, Iori M, Parreño F, Tamarit JM (2017) Mathematical models for multicontainer loading problems. Omega 66:106–117

Alonso MT, Alvarez-Valdes R, Iori M, Parreño F (2019) Mathematical models for multi container loading problems with practical constraints. Comput Ind Eng 127:722–733

Araujo OCB, Armentano VA (2007) A multi-start random constructive heuristic for the container loading problem. Pesqui Operac 27(2):311–331

Araya I, Riff MC (2014) A beam search approach to the container loading problem. Comput Oper Res 43:100–107

Araya I, Guerrero K, Nuñez E (2017) VCS: a new heuristic function for selecting boxes in the single container problem. Comput Oper Res 82:27–35

Battarra M, Monaci M, Vigo D (2009) An adaptive guidance approach for the heuristic solution of a minimum multiple trip vehicle routing problem. Comput Oper Res 36:3041–3050

Beasley J (1985) An exact two-dimensional non-guillotine cutting tree search procedure. Oper Res 33(1):49–64

Bischoff EE, Ratcliff MSW (1995) Issues in the development of approaches to container loading. Omega 23(4):377–390
Bortfeldt A (2012) A hybrid algorithm for the capacitated vehicle routing problem with three-dimensional loading constraints. Comput Oper Res 39:2248–2257
Bortfeldt A, Gehring H (2001) A hybrid genetic algorithm for the container loading problem. Eur J Oper Res 131(1):143–161
Bortfeldt A, Wäscher G (2013) Constraints in container loading. A state of the art review. Eur J Oper Res 229(1):1–20
Bortfeldt A, Gehring H, Mack D (2003) A parallel tabu search algorithm for solving the container loading problem. Parallel Comput 29(5):641–662
Boschetti M, Mingozzi A (2002) New upper bounds for the two-dimensional orthogonal non-guillotine cutting stock problem. IMA J Manag Math 13(2):95–119
Ceschia S, Schaerf A (2013) Local search for amulti-drop multi-container loading problem. J Heuristics 19:275–294
Christensen SG, Rousøe DM (2009) Container loading with multi-drop constraints. Int Trans Oper Res 16:727–743
Christofides N, Whitlock C (1977) An algorithm for towo-dimensional cutting problems. Oper Res 25(1):30–44
Codas A, Camponogara E (2012) Mixed-integer linear optimization for optimal lift-gas allocation with well-separator routing. Eur J Oper Res 217(1):222–231
Correcher JF, Alonso MT, Parreño F, Alvarez-Valdes R (2017) Solving a large multicontainer loading problem in the car manufacturing industry. Comput Oper Res 82(1):139–152
Côté J, Iori M (2018) The meet-in-the-middle principle for cutting and packing problems. INF J Comput 30(4):646–661
Dabia S, Ropke S, Van Woensel T (2019) Cover inequalities for a vehicle routing problem with time windows and shifts. Transp Sci 53(5):1354–1371
Davies AP, Bischoff EE (1999) Weight distribution considerations in container loading. Eur J Oper Res 114(3):509–527
De Almeida Cunha JG, de Lima VL, de Queiroz TA (2019) Grids for cutting and packing problems: a study in the 2D knapsack problem. 4OR-Q J Oper Res. https://doi.org/10.1007/s10288-019-00419-9
Egeblad J, Garavelli C, Lisi S, Pisinger D (2010) Heuristics for container loading of furniture. Eur J Oper Res 200(3):881–892
Eley M (2003) A bottleneck assignment approach to the multiple container loading problem. OR Spectr 25:45–60
EUROSTAT (2017) Eurostat statistics explained, transport. https://ec.europa.eu/eurostat/. Online Accessed 26 Dec 2019
Fanslau T, Bortfeldt A (2010) A tree search algorithm for solving the container loading problem. INF J Comput 22(2):222–235
Fekete SP, Schepers J, Van der Veen JC (2007) An exact algorithm for higher-dimensional orthogonal packing. Oper Res 55(3):569–587
Fuellerer G, Doerner K, Hartl RF, Iori M (2010) Metaheuristics for vehicle routing problems with three-dimensional loading constraints. Eur J Oper Res 201:751–759
Gehring H, Bortfeldt A (2002) A parallel genetic algorithm for solving the container loading problem. Int Trans Oper Res 9(4):497–511
Gendreau M, Iori M, Laporte G, Martello S (2006) A tabu search algorithm for a routing and container loading problem. Transp Sci 40:342–350
Jamrus T, Chien CF (2016) Extended priority-based hybrid genetic algorithm for the less-than-container loading problem. Comput Ind Eng 96:227–236
Jin Z, Ohno K, Du J (2004) An efficient approach for the three-dimensional container packing problem with practical constraints. Asia-Pac J Oper Res 21(3):279–295
Junqueira L, Morabito R, Sato Yamashita D (2012a) MIP-based approaches for the container loading problem with multi-drop constraints. Ann Oper Res 199(1):51–75
Junqueira L, Morabito R, Sato Yamashita D (2012b) Three-dimensional container loading models with cargo stability and load bearing constraints. Comput Oper Res 39(1):74–85
Lim A, Ma H, Qiu C, Zhu W (2013) The single container loading problem with axle weight constraints. Int J Prod Econ 144(1):358–369
Liu WY, Yue Y, Dong Z, Mapple C, Keech M (2011) A novel hybrid tabu search approach to container loading. Comput Oper Res 38:797–807

Mack D, Bortfeldt A, Gehring H (2004) A parallel hybrid local search algorithm for the container loading problem. Int Trans Oper Res 11(5):511–533

Makarem OC, Haraty RA (2010) Smart container loading. Comput Methods Sci Technol 10:S231–S245

Martello S, Pisinger D, Vigo D (2000) The three-dimensional bin packing problem. Oper Res 48(2):256–267

Moura A, Oliveira JF (2005) A GRASP approach to the container-loading problem. IEEE Intell Syst 20(4):50–57

Paquay C, Schyns M, Limbourg S (2016) A mixed integer programming formulation for the three-dimensional bin packing problem deriving from an air cargo application. Int Trans Oper Res 23(1):187–213

Parreño F, Alvarez-Valdes R, Tamarit JM, Oliveira JF (2008) A maximal-space algorithm for the container loading problem. INF J Comput 20(3):412–422

Parreño F, Alvarez-Valdes R, Oliveira JF, Tamarit JM (2010) Neighborhood structures for the container loading problem: a VNS implementation. J Heuristics 16:1–22

Pollaris H, Braekers K, Caris A, Janssens G, Limbourg S (2016) Capacitated vehicle routing problem with sequence-based pallet loading and axle weight constraints. EURO J Transp Logist 5:231–255

Ramos AG, Oliveira JF, Gonçalves JF, Lopes MP (2015) Dynamic stability metrics for the container loading problem. Transp Res Part C Emerg Technol 60:480–497

Ramos AG, Oliveira JF, Lopes MP (2016) A physical packing sequence algorithm for the container loading problem with static mechanical equilibrium conditions. Int Trans Oper Res 23(1–2):215–238

Ramos A, Silva E, Oliveira J (2018) A new load balance methodology for container loading problem in road transportation. Eur J Oper Res 266(3):1140–1152

Ren J, Tian Y, Sawaragi T (2011) A tree search method for the container loading problem with shipment priorities. Eur J Oper Res 214:526–535

Scheithauer G (1992) Algorithms for the container loading problem. In: Gaul W, Bachem A, Habenicht W, Runge W, Stahl WW (eds) Operations research proceedings. vol 1991. Springer, Berlin, Heidelberg, pp 445–452

Scheithauer G, Terno J (1996) The g4-heuristic for the pallet loading problem. J Oper Res Soc 47(4):511–522

Shebalov S, Park Y, Klabjan D (2015) Lifting for mixed integer programs with variable upper bounds. Discrete Appl Math 186(1):226–250

Sheng L, Xiuqin S, Changjian C, Hongxia Z, Dayong S, Feiyue W (2017) Heuristic algorithm for the container loading problem with multiple constraints. Comput Ind Eng 108:149–164

Toffolo TAM, Esprit E, Wauters T, Vanden Berghe G (2017) A two-dimensional heuristic decomposition approach to a three-dimensional multiple container problem. Eur J Oper Res 257(2):526–538

UNCTAD (2018) Report of maritime transport 2018. https://unctad.org. Online Accessed 26 Dec 2019

Wang N, Lim A, Zhu W (2013) A multi-round partial beam search approach for the single container loading problem with shipment priority. Int J Prod Econ 145:531–540

Wäscher G, Haußner H, Schumann H (2007) An improved typology of cutting and packing problems. Eur J Oper Res 183(3):1109–1130

Zhu W, Oon W, Lim A, Weng Y (2012) The six elements to block-building approaches for the single container loading problem. Appl Intell 37(3):1–15