4OR (2020) 18:49–72
https://doi.org/10.1007/s10288-018-0397-z

RESEARCH PAPER

A GRASP algorithm for multi container loading problems with practical constraints

M. T. Alonso[1] · **R. Alvarez-Valdes**[2] · **F. Parreño**[1]

Received: 22 May 2018 / Revised: 28 November 2018 / Published online: 1 January 2019

Abstract
We consider the multicontainer loading problem of a company that has to serve its customers by first putting the products on pallets and then loading pallets onto trucks. When a large number of units of a product have to be shipped, the company requires that homogeneous pallets, with only one product, are built first, then weakly heterogeneous pallets, in which each layer corresponds to a single product, and finally strongly heterogeneous pallets with the remaining units of the products. To be useful in practice, the solutions have to satisfy five types of constraints: geometric constraints, so that pallets are completely inside the trucks and do not overlap; weight constraints, limiting the total weight a truck can bear and the maximum weight supported by each axle; constraints limiting the position of the centre of gravity of the cargo; dynamic stability constraints, to avoid cargo displacement when the truck is moving; and constraints ensuring that the delivery dates of products are respected. We have developed a Greedy Randomized Adaptive Search Procedure, including some improvement methods tailored to the problem, among them an adaptation of ejection chains. The approach has been tested on a benchmark of real problems and it has been shown to be capable of finding high-quality, realistic solutions in short computing times. We also provide a comparison with an integer programming formulation that justifies the use of a metaheuristic algorithm.

Keyword Container loading · Optimization · Heuristics algorithm · GRASP

Mathematics Subject Classification 90B06 (Transportation and logistics)

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

R. Alvarez-Valdes
ramon.alvarez@uv.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, Burjassot, Valencia, Spain

1 Introduction

The Container Loading Problem (CLP) is one of the most extensively studied problems in the field of Cutting and Packing, due to its many practical applications. Given a set of boxes to be transported by trucks (or containers), the problem is how to pack the boxes into the trucks so as to maximize the occupied volume. Loading trucks to full capacity has not only clear economic advantages for the companies involved, but also global environmental advantages derived from the reduction of traffic and its pollutant emissions. When the number of products to be sent exceeds the truck capacity, more than one truck has to be used, and the problem becomes a Multi Container Loading Problem (MCLP).

Both problems have geometric constraints, preventing boxes from overlapping and exceeding the truck's dimensions, but there are also many practical constraints to be satisfied. Concerning weight, not only the total weight but also its distribution on the axles and the position of the centre of gravity of the cargo have to satisfy given limits. There are also stability constraints, static stability when the vehicle is not moving and dynamic stability when the vehicle is moving and subjected to acceleration and braking forces. These practical constraints are required for the solutions to the problems to be useful in practice and have received increased attention in recent years.

The standard CLP and MCLP problems consider that the dimensions of the boxes are given and the boxes are loaded directly into the trucks. However, this is not the case in many practical situations in which first the boxes have to be put onto pallets and then these pallets have to be loaded into trucks. In some cases, the pallets are homogeneous, composed of boxes of just one product, and the problem is simplified, but in others, boxes of many different dimensions have to be combined to form heterogeneous pallets. In this study we consider the problem of a big distribution company that has to move large quantities of products between its central and regional logistics centres. As the quantities to be sent are very large, it is a point-to-point transportation problem usually involving several trucks from each origin to each destination, not requiring any routing. The demanded products have a delivery date within a planning horizon, usually 1 week. Products cannot be received later than their delivery date, but can be received earlier, if that helps to fill up a truck partially occupied by products for a previous day. This prevents us from solving the problem for each day separately, because planning all the days in the horizon jointly can reduce the total number of trucks required.

The composition of pallets is subject to some rules, aimed at simplifying their handling at the warehouses. For each product, the company has decided the composition of the layer, a two-dimensional arrangement of boxes of the product that matches the dimensions of the pallet base with a small tolerance. Whenever possible, this layer structure defined for each product has to be respected. Furthermore, if the demand for a product is high enough to build a homogeneous pallet in which all the layers are composed of the same product, this is the preferred option. These pallets, known as *stock pallets*, are very easy to handle and store. Once the stock pallets have been built, the remaining layers are used to build *case pallets*, which can be seen as weakly heterogeneous, because each layer consists of a single product, although not all the layers correspond to the same product. Finally, if the number of demanded units of a product is not a multiple of the number of units composing a layer, the remaining units are packed into strongly heterogeneous pallets, called *rest pallets*. This pallet structure is very general and can be adapted to the requirements of many other companies that use all or just some of these pallet types.

In this study, we will take the problem of this large company in Europe as a reference, but the problem is common to many other distribution companies around the world. We propose a metaheuristic solution procedure based on a GRASP algorithm, considering not only the general pallet structure described above, but also the practical constraints related to weight and stability that are needed to provide practical solutions. Previous studies on simpler versions of the problem that proposed integer linear models have shown that exact procedures are not adequate to solve large instances in the short times required in practice. Therefore, we have adopted a metaheuristic approach, including constructive procedures, randomization strategies and a set of improvement moves specifically designed for this problem. The computational results on a set of real instances show that the proposed approach is a very efficient way of solving the problems, providing high quality solution in very short computing times.

The structure of the paper is as follows. The relevant literature is reviewed in Sect. 2. Section 3 presents a detailed definition of the problem. The GRASP metaheuristic is presented in Sect. 4 by describing its constructive algorithm, randomization strategies, and improvement moves. Section 5 considers the case in which demands have delivery dates. The computational experience carried out is summarized in Sect. 6. Finally, conclusions and future work are highlighted and discussed in Sect. 7.

2 Related work

Container Loading Problems have been extensively studied due to their important practical applications. Basic geometric constraints define an interesting and complex combinatorial problem for which many exact and heuristic approaches have been proposed. Concerning practical constraints, as early as 1995 Bischoff and Ratcliff (1995) listed conditions to be taken into account when solving practical container loading problems, but in most cases early studies on the CLP did not consider practical conditions, with the result that the solutions were not useful for real world companies. These conditions have been progressively considered in more recent papers. The reviews by Bortfeldt and Wäscher (2013) and Zhao et al. (2016) show the increasing trend toward including these practical constraints in the development of algorithms.

In the problem considered in this study, two conditions are especially important, concerning weight and stability. Many authors include a constraint limiting the maximum weight that can be loaded into the truck [see, for example, Bortfeldt (2012) or Toffolo et al. (2018)] and several authors have also included constraints limiting the weight supported by the axles (Lim et al. 2013; Pollaris et al. 2016; Alonso et al. 2016). Although the usual objective is to maximize the occupied volume, when the cargo is heavy, weight is the more restrictive condition and loading trucks to their maximum weight becomes the most important objective. In addition, the weight of the load has to be evenly spread on the container floor. A good weight distribution would place the centre of gravity near to the geometric centre of the truck, although in general some tolerance is allowed (Baldi et al. 2012; Moon and Nguyen 2014; Queiroz and Miyazawa 2013). Ramos et al. (2018) go further, considering load balance a hard constraint and using the vehicle-specific technical characteristics to satisfy real-world regulations when developing a new genetic algorithm for the CLP.

The other main condition is the stability of the cargo. Static, or vertical, stability ensures the stability of the items being loaded or unloaded when the vehicle is not moving. Usually this type of stability has been ensured by imposing full base support on the items, meaning that their base has to be supported from below by the truck floor or other items (Fanslau and Bortfeldt 2010; Junqueira et al. 2012). However, this approach is too restrictive, even if full support is relaxed to a percentage of the base, as Ramos et al. (2016) have shown. They have developed a more efficient approach to static stability for the three-dimensional case, while Queiroz and Miyazawa (2014) have done the same for the two-dimensional strip packing problem. Dynamic, or horizontal, stability ensures that items will not move when the truck is moving and subjected to acceleration, braking and turns. This type of stability has been less studied. Ramos et al. (2015) propose new metrics, extending the ideas of Bischoff and Ratcliff (1995) and Alonso et al. (2017) consider dynamic stability in their models for the MCLP.

Very few papers consider the problem of packing goods on pallets and then pallets on trucks. According to the typology for cutting and packing problems proposed by Wäscher et al. (2007), the two problems can be classified as Single Stock Size Cutting Stock Problems. Morabito et al. (2000) decompose the problem into two phases. First, a maximum number of products have to be loaded on each pallet. Then, the pallets are loaded in the minimum number of trucks. Both problems are solved by using the 5-block algorithm developed by Morabito and Morales (1998). Takahara (2005) proposes a heuristic in which solutions are represented by an ordered list of items and an ordered list of pallets and containers in which the items have to be loaded. Permutations of these lists, leading to good solutions, are obtained by local search techniques. Sheng et al. (2016) first load pallets into containers but then allow individual boxes to be used to fill the gaps between pallets to achieve a better use of the space. They use a tree search procedure for loading the pallets and then a greedy algorithm for filling the residual spaces. In the problem studied by Alonso et al. (2017), pallets are built by packing horizontal layers that have been previously defined and these pallets have to be loaded into the minimum number of trucks. The authors develop several integer linear models, progressively including more constraints concerning the maximum weight on the axles and the position of the centre of gravity, and test them on a large set of benchmark instances provided by a distribution company. In a parallel study, Alonso et al. (2016) consider the same problem and develop a GRASP algorithm producing good solutions for large instances that could not be solved using integer models.

Problems involving pallet and truck loading also appear when packing and routing are jointly addressed. Combined loading and routing problems first appeared as extensions of the Capacitated Vehicle Routing Problem, so a set of boxes had to be loaded and sent to customers, satisfying two-dimensional (Iori et al. 2007) or three-dimensional packing constraints (Gendreau et al. 2006). Iori and Martello (2010) review the field of routing problems with loading constraints, so here we will focus on those studies in which boxes are put onto pallets before placing them in trucks. Doerner et al. (2007)

deal with a combined routing and packing problem in which the items are placed on pallets and stacked one above the other, producing piles, and propose two metaheuristic algorithms, a Tabu Search, and an ACO algorithm. Zachariadis et al. (2012) also consider a routing and packing problem in which products have to be put into pallets and develop a heuristic strategy for building routes. For each route, the feasibility of the pallet packing is checked by using a set of simple packing rules. Pollaris et al. (2016) combine a vehicle routing problem with the loading of homogeneous pallets. Pallets are placed in two rows and cannot be stacked. They propose a mixed ILP formulation to minimize the transportation cost, respecting the axle weight constraints and the order in which customers are served. Moura and Bortfeldt (2017) also solve a combined vehicle routing and pallet loading problem. In a first phase, they build homogeneous pallets by using the GRASP algorithm by Moura and Oliveira (2005). In a second phase, they load pallets into trucks using a tree search procedure.

The literature review shows that the problem addressed in this paper has not been studied previously. On the one hand, concerning the pallets, some studies only consider homogeneous pallets and most of those considering heterogeneous pallets build them in a first phase before solving the problem of packing them into the trucks. Here we have to take into account three types of pallets, ranging from homogeneous stock pallets to strongly heterogeneous rest pallets, and including weakly heterogeneous case pallets in which each layer is composed of a single product. The way in which the layers are used to build case pallets has to be decided as well as the position of each pallet in the trucks. On the other hand, constraints concerning axle weight distribution and dynamic stability have not been considered yet for the case of different types of pallets. The most closely related studies are those by Alonso et al. (2016) and Alonso et al. (2017), which consider only case pallets and are therefore restricted to the particular situation in which the demand for each product can always be accommodated in an integer number of layers.

3 Problem description

The two main elements of the problem are the products to be served and the vehicles used for delivering them. For each product $j \in J$, the demand for each day d of the planning horizon is n_{dj}. The dimensions of a unit of the product are (l_j, w_j, h_j) and its weight is q_j. The company has also decided beforehand the number of units of the product that compose a layer, l_j, and the number of layers composing a pallet, p_j.

Concerning the trucks, the company uses one type of truck with dimensions (L, W, H), maximum weight Q, and maximum weights supported by the front and rear axles, Q_1 and Q_2. The distances from the front of the truck to the axles are δ_1 and δ_2. The main elements of the truck are shown in Fig. 1 and we assume that the company has a set of available trucks large enough to meet all possible demands.

The company uses one type of pallet of dimensions (l^p, w^p). The number of pallets that fit into the truck is $\lfloor \frac{L}{w^p} \rfloor * \lfloor \frac{W}{l^p} \rfloor$, with $\lfloor \frac{L}{w^p} \rfloor$ pallets along the truck's length and $\lfloor \frac{W}{l^p} \rfloor$ across the truck's width. The set $\mathcal{P}$ of positions of the pallets on the truck floor defines a grid, as can be seen in Fig. 2 for the most common case in which two pallets

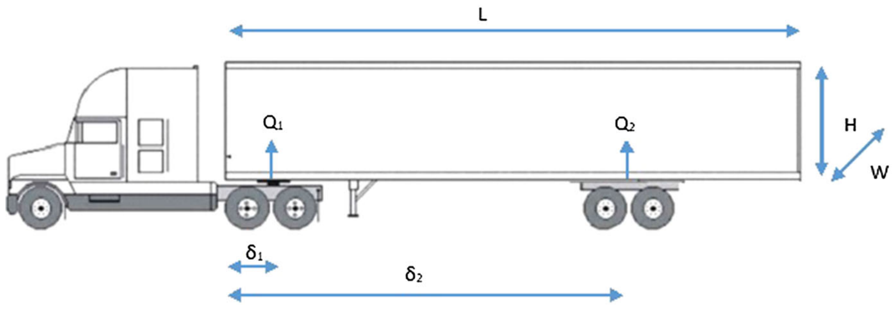


Fig. 1 Dimensions and axles positions of the truck

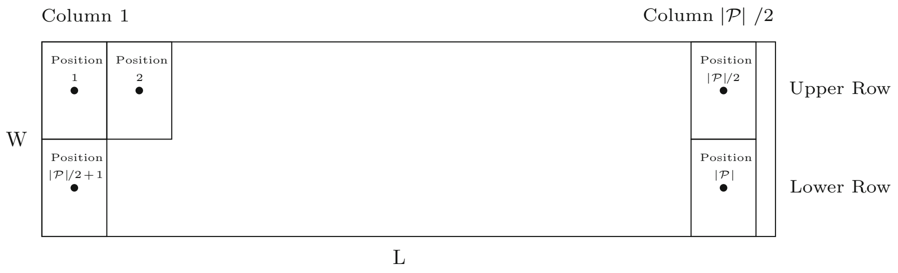


Fig. 2 Pallet positions on the truck floor

fit into the truck width. The $|\mathcal{P}|$ positions are arranged in two rows, with positions 1 to $|\mathcal{P}|/2$ on the upper row and positions $|\mathcal{P}|/2+1$ to $|\mathcal{P}|$ on the lower row, and $|\mathcal{P}|/2$ columns. The positions are defined starting from the front and going to the back of the truck.

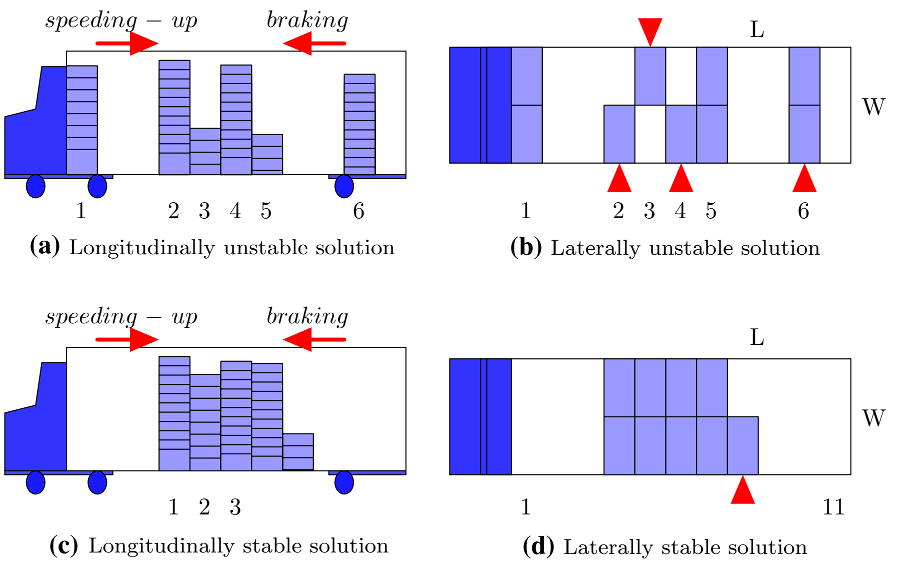
A feasible solution has to satisfy several types of constraints. Besides the basic constraints of having each position on the truck floor occupied by at most one pallet, and ensuring that pallets do not exceed the truck height H, they can be classified in several categories:

3.1 Weight constraints

The total weight of the pallets in a truck cannot exceed Q. The weight on the axles cannot exceed Q_1 for the front axle and Q_2 for the rear axle. The weight supported by each axle depends on the position of the load in the truck and can be calculated if we know the position and weight of each pallet, using the law of levers (Alonso et al. 2017). The centre of gravity of the cargo has to be positioned between the axles.

3.2 Stability constraints

The cargo has to be stable. Static stability when the truck is not moving is ensured by the way in which the pallets are built. Dynamic stability when the truck is moving and subjected to acceleration, braking, and turns has to be ensured when placing the pallets



(a) Longitudinally unstable solution **(b)** Laterally unstable solution **(c)** Longitudinally stable solution **(d)** Laterally stable solution

Fig. 3 Solutions with and without dynamic stability

in the truck. Longitudinal stability, in the length direction of the truck, requires avoiding empty spaces between pallets. Lateral stability, in the width direction, requires having pallets in both rows of the same column. If pallets are placed as in Fig. 3a, pallets 1, 5, and 6 will be displaced towards the back of the truck due to acceleration and pallets 2 and 6 towards the front when the truck brakes. Similarly, if the pallets are placed in the truck as in Fig. 3b, pallets 2, 3, and 4 will be displaced when the truck turns right or left.

Ensuring dynamic stability also requires that the heights of adjacent pallets are not too different, to prevent tall pallets tipping over shorter ones. We impose the condition that pallet heights have to be greater than or equal to the half the height of the truck. As these stability constraints may be too strict in some cases, or even impossible to satisfy, we relax them and allow at most one pallet in each truck with a height lower than half the height of the truck and at most one column with only one pallet. Figure 3c, d show these types of solutions.

3.3 Delivery dates

The delivery dates associated with the demand for the products have to be respected. Products cannot be delivered after their delivery dates, although they can be delivered before. In other words, products ordered for day 1 have to be served on day 1, while products ordered for day 2 can be served on day 2, but also on day 1. Taking the argument to the extreme, all the products ordered for the planning horizon could be delivered on the first day. However, the company does not want to do that, for two main reasons. First, it is advisable to spread the deliveries over the days to balance the number of trucks required each day. Second, sending products throughout the days of the planning horizon makes it easier to accommodate last-minute changes in the orders. Each pallet contains products for 1 day only.

The way in which the company takes advantage of the flexibility in the service is to plan the deliveries in order. First, the products ordered for day 1 are considered and the corresponding problem is solved, producing a solution requiring t_1 trucks. If the last of these trucks is not completely full, as is usually the case, it can be filled with products for day 2, and these t_1 trucks are sent on day 1. Then, the problem involving the products for day 2 that were not sent in advance is solved, producing a solution requiring t_2 trucks. The last of these trucks is filled up with products for day 3, and the process continues until all the days in the horizon have been considered.

The main objective of the problem is to minimize the total number of trucks required to send all the demanded products throughout the planning horizon. A second objective is to minimize the load in the last truck. For two solutions with the same number of trucks, a solution is better if the load in the last truck is lower. This load can be measured by weight or by volume, depending on which of them is the more influential factor in the solution. Minimizing the load of the last truck allows the user to decide between two possibilities. He/she can decide not to send it if it contains just a few units of non-urgent products. Alternatively, if the information is already available, the truck can be filled with products demanded for the next planning horizon, usually the following week.

In order to attain these objectives, we have to take two types of decisions, concerning how to build pallets and how to place them in trucks. First, in a preprocessing phase, some pallets are built according to the company's requirements. For each product j and day d the demand is n_{dj} units and a layer is composed of l_j units, so we will have $L_{dj} = \lfloor n_{dj}/l_j \rfloor$ layers of product j and $r_{dj} = n_{dj} - l_j L_{dj}$ remaining units. With these L_{dj} layers we build stock pallets, homogeneous pallets in which all layers are composed of the same product. As the number of layers in a pallet is p_j, we can build $P_{dj} = \lfloor L_{dj}/p_j \rfloor$ stock pallets of product j for day d, leaving $s_{dj} = L_{dj} - p_j P_{dj}$ remaining layers. The remaining units of the products for a day, $r_{dj}, j \in J$, are then taken to build rest pallets by using a three-dimensional bin packing algorithm developed in a previous study (Parreño et al. 2010). So after this phase we have some stock pallets, some rest pallets, and some layers that have to be combined to build case pallets. The decisions about how to build these pallets and where to place all the pallets built are considered jointly in the GRASP algorithm described in the next section, avoiding the usual two-phase approach in which pallets are built first and their position in the trucks decided later.

4 GRASP algorithm

In this section we describe the elements of the Greedy Randomized Adaptive Search Procedure (GRASP) developed for this problem. First, we introduce the deterministic constructive algorithm, then the randomizing strategies included to produce diverse solutions in the iterative process, and finally the improvement phase. A schematic representation of the complete procedure appears in Algorithm 1. In this section we consider the problem in which all products are served on 1 day. The extension to all days of the planning horizon will be described later.

Algorithm 1 GRASP algorithm

```
$K$= Number of trucks generated by the deterministic constructive algorithm
$N$= Number of iterations used to set the statistical filter
$TimeLimit, NmaxIterations, K, N$
for (Iterations< $NmaxIterations$ or $Time < TimeLimit$) do
    Solution=Generate a Randomized Solution with $K - 1$ trucks
    Statistical filter(Solution)
    if (Solution has passed the filter or Iterations< $N$) then
        Local Search(Solution)
    end if
    if (Solution has all the layers packed) then
        $K = K - 1$
        Re-start statistical filter
    end if
end for
```

GRASP algorithms have been successfully applied to other related packing problems (Correcher et al. 2017; Moura and Bortfeldt 2017) as well as other hard combinatorial problems in routing (Lopez-Sanchez et al. 2018), scheduling (Knopp et al. 2017), or location (Contreras et al. 2017), among others. It provides a flexible algorithmic framework in which constructive and local search procedures tailored to the problem being solved can be combined in an iterative process that includes randomization to provide diversity to the search. The GRASP algorithm developed in this paper shares some basic ideas with that of our previous study (Alonso et al. 2016), especially in the constructive algorithm and randomization strategy, but there are two main differences: the existence of three types of pallets, producing special types of layers and changing the criteria used in the randomized constructive procedure, and the way in which the delivery dates are taken into account when building a complete solution, requiring the development of completely new improvement moves.

4.1 Deterministic constructive algorithm

As explained in Sect. 3, the products ordered have been used to build stock pallets, rest pallets, and a set of layers that will compose case pallets. In what follows, stock and rest pallets are considered as layers, although they are special layers, because the pallets which will be built using them will contain only one of these special layers. This can be ensured by assigning them a height H, to prevent other layers from being stacked above or below them. Hence, the constructive procedures will only consider a set $\mathcal{L}$ of layers as the elements to be packed.

The constructive algorithm loads one truck at a time, using two lists, the list of available positions on the truck floor and the list of the remaining layers to be packed. The main steps of the algorithm are:

- *Step 1. Initialization*
  Let $\mathcal{P}$ be the list of positions on the truck, initially the complete list of positions defined for the truck.
  Let $\mathcal{L}$ be the list of layers not packed in previous trucks. If the first truck is being packed, the list contains all demanded layers (including stock and rest pallets).
  The layers on list $\mathcal{L}$ are ordered first by non-increasing order of height, to give priority to layers composed of stock and rest pallets. As a tie-breaker, we consider non-increasing weight, placing heavy layers first. If necessary, a second tie-breaker is the density of the layer.
- *Step 2. Selecting the position*
  We determine the point in the truck that can accommodate the maximum weight, taking into account the maximum weight the axles can support. As usually $Q_1 > Q_2$, this point is not exactly the geometric centre of the truck, but is slightly displaced towards the front.
  The positions are chosen starting from the position nearest to this maximum-weight point, and moving towards the front or the back, following two rules:

  – The two positions in each column are considered consecutively.
  – The next column to be considered will be one position backward or forward of the columns already occupied and it is chosen depending on the position of the centre of gravity. If it is currently displaced forwards, the next available position further back is chosen, to move the centre of gravity towards its ideal position. If the centre of gravity is displaced backwards, the next position further forward is chosen.

- *Step 3. Building a pallet suitable for the chosen position*
  Once a position has been chosen, we determine the maximum weight that can be placed in that position without exceeding the total weight limit and the maximum weight on the axles.
  A new pallet is built by stacking layers from $\mathcal{L}$ in order, without exceeding the truck height and the maximum weight previously calculated for the pallet.
  If a pallet cannot be built in the selected position because putting any layer there would exceed the maximum weight, no more pallets can be placed in other positions in the truck due to the stability constraints and the truck is closed. If some layers can be placed but the resulting pallet height is lower than half the truck height, it will be the last pallet that can be placed in the truck and the truck is also closed.
- *Step 4. Updating the lists*
  Lists $\mathcal{P}$ and $\mathcal{L}$ are updated with the remaining positions in the truck and the layers remaining to be packed. The position of the centre of gravity and the weights supported by the axles are also recalculated taking the new pallet into account. If $\mathcal{L} = \emptyset$ the packing is complete and the algorithm ends. Otherwise, if $\mathcal{P} = \emptyset$ a new truck is opened with its complete list of positions. The procedure goes back to Step 2.

4.2 Randomization strategies

At Step 2, the selection of the next backward or forward position is randomized, with probabilities proportional to the remaining weight that each axle can support.

We also randomize the selection of the layers being taken to build the pallet at Step 3. We use the *sample plus construction* proposed by Resende and Werneck (2004). A random sample is taken from among all products with layers on list $\mathcal{L}$. The sample size is controlled by a parameter δ, so each product has a probability δ of being part of the sample. We go through the sample, ordered by the above-mentioned criteria of height, weight and density, and the first product with a layer that satisfies all the height and weight constraints is selected for the pallet being built at the chosen position. If the list $\mathcal{L}$ contains more than one layer of the selected product, the number of layers taken for building the pallet is chosen at random. If $\delta = 1$, all products are included in the sample and the first one on the list will always be chosen, making the selection deterministic. If δ is closer to 0, the number of products in the sample is smaller and the first ones on list $\mathcal{L}$ have less probability of being chosen for the sample. Therefore, the layers finally selected may not be the best options for the position according to its height and weight. In this sense, the procedure introduces diversity in the set of solutions obtained by the constructive algorithm. However, it is possible that none of the layers of the products in the sample satisfies all the constraints of the position. If this is the case, we go through the complete list of layers $\mathcal{L}$ and the first layers that satisfy the constraints are chosen.

In order to introduce even more diversity into the solutions, we use one more randomizing procedure. We randomize the order in which the criteria—height, weight, density—are considered. Also, instead of always ordering the layers by non-increasing height, weight, and density, at each iteration we randomly choose between non-increasing and non-decreasing orders.

The randomized constructive procedure as described, packing all the layers into trucks, is only used in the first iteration of the GRASP algorithm. Once we have a feasible solution with K trucks, in the subsequent iterations the objective is to obtain a solution with $K - 1$ trucks. Step 4 is modified and when truck $K - 1$ is closed, the algorithm ends and the solution will be composed of these $K - 1$ trucks and a list of unpacked layers. The improvement phase will try to load these unpacked layers into some of the $K - 1$ trucks.

4.3 Improvement phase

The solution built by the randomized constructive procedure will usually contain some unpacked layers. The objective of the improvement moves is to reduce the total height or weight of these layers, trying to obtain a solution with no unpacked layers. If a solution with no unpacked layers is obtained, we have a new best solution with one fewer truck. On the rare occasions on which the randomized constructive procedure produces a solution in which all layers are packed using one truck fewer than the best solution obtained so far, this solution is not improved and the procedure goes directly to the next iteration, decreasing the target number of trucks by one.

Depending on the characteristics of the instance, we determine whether the more influential factor in the solution is the weight of the products or their volume, and therefore height (given the common surface of all pallets). We compute the average percentage height and the average percentage weight of the pallets for all the trucks in the best solution and the maximum of these values defines the secondary objective function. For example, if we have a problem in which we have few layers in each truck but they are very heavy, the objective will be the weight and we would want to optimize the weight in each truck and to reduce the weight of the unpacked layers. This is evaluated every time a new best solution is found. In order to introduce more diversity, in each iteration we have a 20% probability of choosing randomly if we consider height or weight as the secondary objective.

We have developed four types of improvement movements. The first consists of removing some elements of the solution and filling the truck again using the constructive deterministic algorithm. The second consists of exchanging layers placed in trucks with unpacked layers. The third is based on the idea of changing the position of pallets inside a truck in order to reduce the weight on one axle, leaving room for new pallets. Finally, the fourth movement is an adaptation of the ejection chains to this packing problem.

In the description of the movements, we will consider that the secondary objective function is to minimize the total height of the unpacked layers. A similar argument would be used for minimizing the weight.

4.3.1 Unloading part of one truck

In this movement, we try to increase the load of the trucks whose average height of pallets is lower than the average height of all the pallets in the solution. We choose one of these trucks at random and randomly select a pallet in this truck among those whose height is less than 90% of the truck height. We then have two alternatives: to remove the complete pallet or to remove just some of its upper layers. We assign a probability of 25% to the first option and 75% to the second. If the second option is selected, we choose a value at random, Rh, between 0 and the truck height H, and remove all layers totally or partially contained in the interval (Rh, H). Note that in some cases removing a layer or a pallet can produce a solution that does not satisfy the weight constraints. In these cases we do not remove them, because we always work with feasible solutions.

After removing some layers or a pallet from one truck, we fill it again using the deterministic constructive algorithm, considering the list of unpacked and removed layers ordered by non-increasing height. Nevertheless, when there are few unpacked layers, we can obtain better results by solving a *subset sum problem* (SSP): given the set J of unpacked and removed layers, each with height h_j, and the remaining height H', find the subset of J whose total height is a maximum but does not exceed H'. The SPP can be formally stated by introducing a binary variable ξ_j taking value 1 if and only if layer j is in the selected subset, thus obtaining:

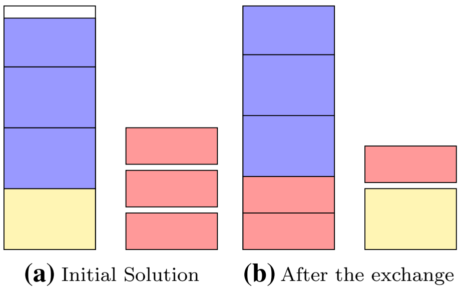


Fig. 4 Exchanging layers

$$\text{(SSP)} \ \max \quad R = \sum_{j \in J} h_j \xi_j \tag{1}$$

$$\sum_{j \in J} h_j \xi_j \leq H' \tag{2}$$

$$\xi_j \in \{0, 1\} \quad j \in J \tag{3}$$

The SPP is solved by using a knapsack algorithm (Pisinger 2000), but only when the total height of the remaining layers is less than ten times the remaining height of the stack. For large numbers of layers, the deterministic constructive algorithm is used.

4.3.2 Exchanging layers

We try to exchange each layer of each pallet of each truck with one or more unpacked layers in order to decrease the total height of the unpacked layers in the solution. The exchange is made only if it produces an improved feasible solution. Figure 4 shows an example. The large rectangle on the left-hand side represents one pallet in a truck and the small red rectangles are unpacked layers. The yellow layer is exchanged with two red layers, reducing the total height of the layers outside the trucks.

As there are no specific selection criteria, the order in which the layer, the truck, and the position are selected is randomized to add diversity to the search. After making the exchange, as the weights on the axles of the truck may have been reduced, we try to pack more layers with the deterministic version of the constructive algorithm.

4.3.3 Levelling the weight of the truck

Sometimes the randomized constructive algorithm produces trucks in which the load is unbalanced and the centre of gravity of the cargo is displaced towards the back of the truck. It may happen then that the limit of the weight on the rear axle, which is usually lower than the limit on the front axle, is reached and no more pallets can be placed in the truck, but there is some margin in the weight supported by the front axle. The improving movement consists of changing the position of one pallet to an empty

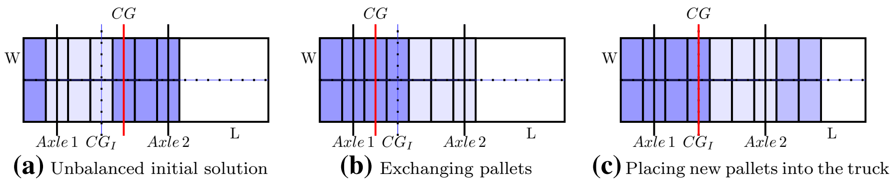

CG
W
L
$Axle\,1$
CG_I
$Axle\,2$
(a) Unbalanced initial solution
(b) Exchanging pallets
(c) Placing new pallets into the truck

Fig. 5 Levelling the load of a truck

space on the other side of the truck or exchanging the positions of two pallets in order to get a more balanced weight distribution that makes it possible to load new pallets.

In Fig. 5a, we can see that CG, the centre of gravity of the load, is displaced to the right of CG_I, the point which can bear more load in the truck, because light pallets (in light blue) are placed to the front and heavy pallets (in dark blue) are placed to the back of the truck. The weight limit on the rear axle is reached and no more pallets can be loaded although there are still some empty positions.

By exchanging the positions of light and heavy pallets, the centre of gravity is moved to the left. The weight on the front axle increases, without exceeding its limit, and the weight on the rear axle decreases (Fig. 5b). Sometimes this makes it possible to place new pallets in the empty spaces at the back of the truck. We check this possibility by again applying the deterministic constructive algorithm, considering the list of the empty positions in the truck and the list of unpacked layers (Fig. 5c).

4.3.4 Ejection chains

An ejection chain movement starts from a feasible solution by selecting an element to undergo a change (Glover 1996; Peng et al. 2016). That produces a new structure, called a *reference structure*, which is similar to a solution but is not feasible, because some constraint is violated. Moves transforming a solution into a reference structure, or one reference structure into another, are called *ejection moves*. Other types of moves, transforming a reference structure into a solution, are called *trial moves*. An ejection chain is composed of ejection moves and trial moves.

In our problem, we use two types of ejection moves. The first moves one unpacked layer to a pallet placed in a truck. This move produces an unfeasible solution, a reference structure in the terminology of ejection chains, because if this layer could go into this pallet without violating any constraint, it would have been put there by the constructive algorithm. Usually the maximum height of the pallet is exceeded, although some other constraints related to weight may also be violated. The other ejection move changes the position of one layer to another pallet in the same truck, transforming one reference structure into another. There are also two types of trial moves. One type removes layers from a pallet and leaves them unpacked until feasibility is recovered. The other moves layers from one pallet to another, recovering the feasibility of the first pallet without making the second unfeasible.

An example can be seen in Fig. 6. In the initial feasible solution we have two pallets (big rectangles) in a truck and two red unpacked layers (Fig. 6a). A first ejection move takes one unpacked layer and moves it to the second pallet, producing a reference

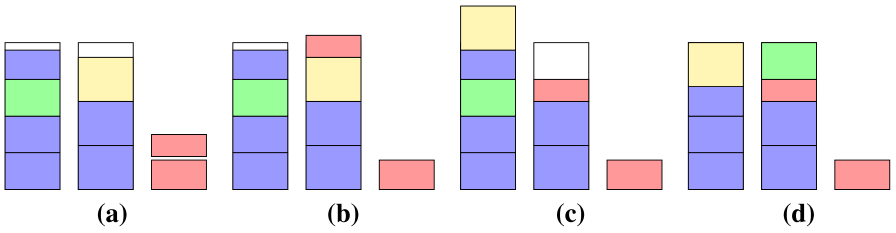


Fig. 6 An example of ejection chains

structure (Fig. 6b). A second ejection move changes the yellow layer to the first pallet, producing a new reference structure (Fig. 6c). Finally, a trial move changes the green layer to the second pallet, producing an improved feasible solution (Fig. 6d).

As in previous movements, we randomize the order in which we select the unpacked layer, the truck, the pallet and the layers in that pallet in order to add diversity to the search.

4.3.5 Local search

The four improvements can be used in a sequential way, following the order in which they have been described, using each of them while the solution is improved, going to the next when it is not, and finishing when the fourth movement does not improve the solution further.

However, the improvement movements can also be combined in a local search procedure. We apply each movement once and after applying all of them, if the solution has been improved, we repeat the procedure.

The complete flow of the local search procedure appears in Algorithm 2.

4.3.6 Statistical filter

Local search is often the phase of a GRASP algorithm requiring the largest computational effort. A way of reducing this effort is to apply a filter. Low-quality solutions built by the randomized constructive algorithm that are unlikely to improve the best current solution after applying local search are discarded without proceeding to the improvement phase. This strategy is called GRASP filtering (Feo et al. 1994).

In order to obtain a filter, during the first N iterations of GRASP all solutions go to the improvement phase. We save their objective function before and after the local search and calculate the average improvement of the secondary objective function. In subsequent iterations, a solution produced by the randomized constructive phase goes into the improvement phase only if its secondary objective function minus the calculated average improvement is lower than the value of the best current solution. Every time the number of trucks required is decreased, this average is recalculated using the next N iterations.

Algorithm 2 Local Search

```
Ensure: $Improvements \leftarrow (1, 2, 3, 4)$ $S : Solution$, $O_s : Objectivefunction$
  while Stopping Criterion is not satisfied do
    $\{S, O_s\}$
    while ImproveSolution==true do
      $ImproveSolution = false$
      for $i \in Improvements$ do
        switch (Improvements[i])
        case 1:
        $\{S, O_s\} \leftarrow ImproveRemoving()$
        if Solution has improved then
          $ImproveSolution = true$
        end if
        case 2:
        $\{S, O_s\} \leftarrow ImproveExchanging()$
        if Solution has improved then
          $ImproveSolution = true$
        end if
        case 3:
        $\{S, O_s\} \leftarrow ImproveLevelling()$
        if Solution has improved then
          $ImproveSolution = true$
        end if
        case 4:
        $\{S, O_s\} \leftarrow ImproveEjectionChain()$
        if Solution has improved then
          $ImproveSolution = true$
        end if
        end switch
      end for
    end while
  end while
```

5 Demand over time

In some practical situations delivery dates are associated with product demands. Some products have to be received on day 1, some others on day 2, and so on, within a planning horizon which usually corresponds to the working days of a week. As previously explained, the way in which delivery dates are set allows the products to be sent before their delivery date, but not later. Therefore, if there is some space left in the trucks carrying the products for the first day, it can be used to load products for the second day until the trucks are completely filled.

Products with different delivery dates can be together in a truck, but the day structure has to be maintained. If in truck k_0 there are products with delivery date d_0, in the following trucks $k > k_0$ we cannot load products with earlier delivery dates $d < d_0$. For example, if products with delivery date $d = 1$ need two complete trucks and part of a third truck, then trucks 1 and 2 will only contain products with delivery date 1, and truck 3 will contain the remaining products for day 1 plus some products for day 2.

The GRASP algorithm described in the previous section has been adapted to the case of delivery dates in the following way: In the first iteration, we start running the randomized constructive algorithm only for the products for day 1. After that we do the same for day 2, starting from the last truck used for day 1, and so on. Then we have a number of trucks used for products for day 1 (K_1), a number for products for days 1 and 2 (K_2), and so on (K_d). For the following iterations, we run the randomized constructive algorithm for day 1, but only for a number of trucks $K_1 - 1$. If we obtain a solution with all these products in $K_1 - 1$ trucks, we update the value of K_1. Otherwise, we have some unpacked layers of products for day 1, which are loaded into a new truck with the deterministic constructive algorithm. Then we apply local search to reduce or even eliminate the layers for day 1 on truck K_1. After that, we start from this partially loaded truck considering the products for day 2, and we repeat the process until the end of the planning horizon.

Applying the local search to the partial solution of day 1 follows exactly the procedure described in the previous section. Nevertheless, when applying it to the solution involving other days, the day structure has to be maintained. If the first movement, removing and refilling, is applied to a truck containing products for two different days and products for the first of these 2 days are removed, these products are always considered the first when loading the truck again. The second move, exchanging unpacked and packed layers, is only applied to products for the same day. The third move, levelling, only involves pallets in one truck and can always be applied. The fourth move, ejection chains, again involving unpacked and packed layers, is applied only to products for the same day. The statistical filter is only applied to the solution obtained by the constructive algorithm for the products with the latest delivery date. All partial solutions obtained by the randomized constructive algorithm go to the improvement phase, with one exception: if the best current solution for products up to day r requires K_r trucks plus some unpacked layers for day r, and the solution being built requires $K_r + 1$ plus some unpacked layers, it is discarded and we proceed to the next iteration.

Algorithm 3 GRASP for Days

```
1: $MaxIter$= Maximum number of Grasp iterations
2: for $Iter = 1$ to $MaxIter$ do
3:     for $d = 1$ to $Days$ do
4:         Randomized Constructive algorithm, starting from the last truck used in the previous day
5:         Local Search on the layers of the last truck
6:         Try to pack some layers of day d+1 in the last truck of day d if it is not the last day
7:     end for
8: end for
```

6 Computational results

The benchmark used in this study is composed of 111 real instances taken from the everyday distribution activity of a large company. The instances have been provided to us by ORTEC (2018), a company developing planning and optimization solutions and services for manufacturing and logistics companies, and are available at the ESICUP web page (https://www.euro-online.org/websites/esicup/data-

Table 1 Characteristics of the test instances

	Products	Units	Stock pallets	Layers	Rest pallets
Minimum	1	241	10	145	0
Median	8	23,904	54	1004	1
Average	13.6	27382.0	70.7	1183.6	1.1
Maximum	142	114,570	299	4873	6

sets/#1535975694118-eedb4714-39e4). The same instances were used by Alonso et al. (2016, 2017) in previous papers, labeled Demand over the time. The instances exhibit wide variability. The distribution of the products ranges between 1 and 142 different types. The demands vary from 241 to 114,570 units. Nevertheless, in order to ensure that the three types of pallets, stock, case, and rest, were sufficiently represented, the original instances were slightly modified. The new instances are available at (https://www.euro-online.org/websites/esicup/data-sets/#1535975694118-eedb4714-39e4), labeled 4OR_GRASP_MC. The characteristics of the instances are summarized in Table 1. The table includes minimum, maximum, median, and average of the number of products, number of units, number of stock pallets built, number of layers after building stock pallets, and number of rest pallets built with remaining units not composing layers. Although in some cases there are no rest pallets, in most cases a bin packing algorithm had to be used to compose rest pallets.

The computational study has four parts. First, we study the contribution of each element of the GRASP algorithm (constructive procedure, randomization, improvements, local search, filtering) to the quality of the solutions obtained. Then, we study the contribution of each type of improvement move in more detail. In a third part, we study the effect of the stability constraints and the particular way in which the company wants the pallets to be built. Finally, we compare GRASP with the integer linear model developed in Alonso et al. (2017) for a closely related problem in which neither stock pallets nor rest pallets are considered and each product makes up an integer number of layers for building case pallets.

The tests of the GRASP algorithm were performed on a PC Intel Core 2 Quad at 2.93 Ghz with 4 GB of RAM in one thread. The integer linear model was solved on the same computer using CPLEX 12.51 with 4 threads.

6.1 Studying the elements of the GRASP algorithm

Table 2 shows the number of trucks in the solutions obtained by progressively adding the elements of the GRASP algorithm:

- The *Constructive* deterministic algorithm.
- The *Randomized* constructive algorithm, in which the value of the parameter δ is randomly chosen in the interval (0.25, 0.75) at each iteration.
- The GRASP algorithm when each improvement move is used just once in the *Improvement* phase, in the order described in Sect. 4.3.

Table 2 Studying the effect of the elements of GRASP

	Deterministic	Randomized	Improvement	Local Search	Filter	Time
1 Day	1151	1146	1118	1117	1117	25.67
2 Days	1152	1147	1118	1117	1117	34.09
3 Days	1153	1148	1119	1118	1118	39.93
Average time	0.01	2.79	11.67	50.52	33.23	33.23

- The GRASP algorithm using *Local Search* in the improvement phase.
- The GRASP algorithm using the filter to select the solutions to be improved (*Filter*).

We set a maximum number of 700 iterations for all the versions with one delivery date and 500 iterations when there is more than one delivery date, in order to have similar computing times.

Table 2 also shows the effect of imposing different delivery dates for the products being served. If not all the products are sent on the same day, the problem is less flexible and the number of trucks required can increase. The table shows three cases, for one, two, and three delivery dates. Columns 2–6 show the total number of trucks required by the 111 test instances. The last row contains the average running times, while the last column contains the average time for solving all versions of the algorithms, showing the progressive complexity added by the delivery dates.

It can be seen in Table 2 that randomizing the constructive procedure has a small effect on the quality of the solutions, while the improvement phase produces a clear decrease in the number of trucks required. The more complex Local Search does not produce a clear effect concerning the number of trucks and the filtering strategy reduces the running time without worsening the solutions.

However, the number of trucks required is not the only criterion for measuring the quality of the solution. For two solutions with the same number of trucks, a solution is better if the last truck is less filled, in weight or volume. Table 3 shows the average percentage in weight and volume of all the trucks but the last in solutions with the same number of trucks for three versions of the GRASP algorithm. Solutions with different number of trucks cannot be compared because if the number of trucks is reduced the last truck will be filled almost to its full capacity, while in other solutions with one more truck the last truck can be almost empty. Column 8, *Same Value*, of the table indicates the number of instances for which the number of trucks is the same for the three versions. The last two columns show the number of instances in which the occupied volume and total weight in the last truck is lower than the 10% and 20% of the capacity. Although the differences are quite small, it can be seen that the version of GRASP using Local Search gets slightly better results, that is, it is able to leave the last truck emptier. When the last truck is almost empty, with less that 10% or 20% of its capacity used, the user has to decide whether (a) to send the truck as it is, (b) not to send it, holding the products back until the next plannign horizon, or (c) to fill the rest of the truck with products which will be demanded in the subsequent periods.

Table 3 Comparing different algorithms in instances with the same number of trucks

	Improvement		Local search		Filter		Same	Last truck	
	Weight	Height	Weight	Height	Weight	Height	Value	$< 10\%$	$< 20\%$
1 Day	93.80	93.98	94.15	94.27	94.08	94.21	108	14	28
2 Days	93.75	93.96	94.05	94.18	93.90	94.04	110	13	30
3 Days	93.89	94.09	93.79	94.03	93.77	94.01	110	13	28

Table 4 Number of times each move obtains the best solution

	Randomized	Remove part	Levelling	Exchange	Ejection chains
1 Day	17	25	11	21	37
2 Days	15	26	8	30	32
3 Days	17	31	12	22	29

6.2 Studying the effect of the improvement moves

The improvement phase of the GRASP algorithm is composed of four improving moves that work on the solution provided by the randomized constructive procedure. Table 4 studies the contribution of each move to the quality of the solutions. For the 111 test instances, the table shows how many times the best solution was obtained by the randomized constructive algorithm and how many times it was obtained when applying the moves in the order in which they appear in the columns of the table. It is remarkable that all moves contribute to obtaining the best solutions, even if they are applied after other moves, indicating that they explore different regions of the solution space. In particular, the adaptation of the ejection chains, although applied at the end of the improvement phase, is able to improve a significant number of solutions, confirming the power of its complex structure for obtaining solutions that simpler moves are not able to achieve.

6.3 Studying the effect of constraints

In this part of the computational study we consider four variants of the problem. The problem described in Sect. 3 considers three types of pallets, stock, case, and rest pallets, with stability constraints, and is refered to as *Stock Stable*. The version in which the three types of pallets are considered but the stability constraints are relaxed is called *Stocks*, and the version in which we consider stability constraints but layers are taken individually, not building stock pallets, is called *Stable*. Finally, the most relaxed version, with neither stability nor stock pallets, is called *Unconstrained*. Comparing these four versions will allow us to assess the influence of each factor on the number of trucks required.

Table 5 shows that although imposing the company's condition of building stock pallets whenever possible makes the problem less flexible, it does not significantly

Table 5 Different variants of the problem

	Unconstrained	Stable	Stocks	Stocks stable
1 Day	1107	1111	1111	1117
2 Days	1109	1112	1111	1117
3 Days	1110	1112	1111	1118

Table 6 Comparing with an integer linear model

	Integer model				GRASP			GRASP versus IP		
	Opt.	No sol.	Time	Trucks	Opt.	Time	Trucks	=	<	>
Unconstrained	103	0	277.1	1109	105	48.1	1107	109	2	0
Stable	105	0	230.3	1112	106	47.5	1111	110	1	0
Stocks	99	5	402.4	1118	102	33.0	1111	99	9	3
Stocks stable	93	11	474.5	1122	95	25.7	1117	101	8	2

increase the number of trucks required. Similarly, the stability constraints severely limit the space of the feasible solutions, but this is not reflected in a dramatic increase in the number of trucks. In conclusion, these practical constraints, which are absolutely necessary in order to obtain useful solutions, do not significantly increase the cost of the solutions, although they do increase the complexity of the solution procedures.

6.4 Comparing with an integer model

The model proposed by Alonso et al. (2017) considers that products are already grouped into layers and that the pallets have to be built from the demanded set of layers and then loaded into trucks. Therefore, for all four versions of the problem defined above, the rest pallets built by using a bin packing algorithm have to be considered as layers of height H so as to prevent other layers being stacked above or below them. For the versions of the problem considering stock pallets, each of these stock pallets has to be considered as an individual layer of height H. Therefore, a product $j \in J$ with a demand of n_{dj} units, composing $L_{dj} = \lfloor n_{dj}/l_j \rfloor$ layers, and with these layers $P_{dj} = \lfloor L_{dj}/p_j \rfloor$ stock pallets, will be decomposed into two products, j_1 with P_{dj} layers of dimensions (l_j, w_j, H) and weight $p_j * l_j * q_j$, and j_2 with $L_{dj} - \lfloor L_{dj}/p_j \rfloor$ layers of dimensions (l_j, w_j, h_j) and weight $l_j * q_j$. As the number of products can be almost doubled, these versions of the problem will be harder to solve for the integer linear model.

Table 6 shows the comparison between the integer linear model by Alonso et al. (2017) and the proposed GRASP algorithm. The integer model has been solved using CPLEX 12.51 with a time limit of 3600 s in four threads. The four columns corresponding to the integer program show the number of optimal solutions, the number of instances for which no feasible solution was found within the time limit, the average running time in seconds and the total number of trucks required by the 111 instances. When a feasible solution was not found, the solution of a simple constructive heuristic was provided. It can be seen that the use of stock pallets, duplicating products, makes the model more difficult to solve. The number of optimal solutions decreases, for some instances no solution was found, and the running time increases as well as the number of trucks. Similarly, the three columns corresponding to GRASP show the number of optimal solutions, the average running time, and the total number of trucks required. It can be observed that GRASP is able to obtain most of the optimal solutions in much shorter running times and with fewer trucks than the integer model. The last three columns in Table 6 compare GRASP and IP on each of the 111 instances for the four variants of the problem, indicating the number of times GRASP obtains the same (column "="), a better (column ">"), or a worse solution (column ">") than the integer model. GRASP obtains equal or better solutions in all but 5 cases of the 444 comparisons. Moreover, the integer model fails to obtain a feasible solution 16 times, thus reducing its practical application.

7 Conclusions

Logistics companies, serving large quantities of products to their clients, involving many trucks, require fast and high-quality solutions that minimize the number of trucks required while satisfying many practical constraints, especially concerning weight and dynamic stability.

To solve this large-scale and highly constrained multicontainer loading problem, we have designed a meta-heuristic GRASP algorithm. We have developed a new constructive algorithm, adapting procedures which were successful for the container loading problem, and have defined some new improvement moves, specific to the problem, including an adaptation of ejection chains.

An extensive computational experiment using a benchmark of real problems provided by the company shows that the GRASP algorithm produces solutions that are optimal or very near to optimality for most of the instances considered in the study, with different characteristics. Some of these characteristics, such as stability, dealing with delivery dates, or minimizing the load on the last truck, are very difficult to implement with mathematical models.

Finally, we would like to mention that the algorithm is quite flexible and could be adapted to accommodate other variants of the problem, including, for example, the possibility of having a fleet of heterogeneous vehicles. Logistics constraints, such as imposing the condition that a subset of the products must be on the same truck, or the opposite, forbidding some classes of products from being on the same vehicle, can also be included.

Acknowledgements This study has been partially supported by the Spanish Ministry of Science and Technology DPI2014-53665-P and by Consejeria de Educacion y Ciencia, Junta de Comunidades de Castilla-La Mancha SBPLY/17/180501/000282.

References

Alonso MT, Alvarez-Valdes R, Parreño F, Tamarit JM (2016) Algorithms for pallet building and truck loading in an inter-depot transportation problem. Math Probl Eng. Article ID 3264214

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

Baldi MM, Perboli G, Tadei R (2012) The three-dimensional knapsack problem with balancing constraints. Appl Math Comput 218:9802–9818

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(9):2248–2257

Bortfeldt A, Wäscher G (2013) Constraints in container loading. A state of the art review. Eur J Oper Res 229(1):1–20

Contreras I, Tanash M, Vidyarthi N (2017) Exact and heuristic approaches for the cycle hub location problem. Ann Oper Res 258:655–677

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

Doerner KF, Fuellerer G, Gronalt M, Hartl RF, Iori M (2007) Metaheuristics for the vehicle routing problem with loading constraints. Networks 49(4):294–307

Fanslau T, Bortfeldt A (2010) A tree search algorithm for solving the container loading problem. INFORMS J Comput 22(2):222–235

Feo T, Resende MGC, Smith SH (1994) A greedy randomized adaptive search procedure for maximum independent set. Oper Res 42(5):860–878

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

Glover F (1996) Ejection chains, reference structures and alternating path methods for traveling salesman problems. Discrete Appl Math 65:223–253

Iori M, Martello S (2010) Routing problems with loading constraints. TOP 18(1):4–27

Iori M, Salazar González JJ, Vigo D (2007) An exact approach for the vehicle routing problem with two-dimensional loading constraints. Transp Sci 41:253–264

Junqueira L, Morabito R, Yamashita DS (2012) Three-dimensional container loading models with cargo stability and load bearing constraints. Comput Oper Res 39(1):74–85

Knopp S, Dauzere-Peres S, Yugma C (2017) A batch-oblivious approach for complex job-shop scheduling problems. Eur J Oper Res 263:50–61

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

Lopez-Sanchez AD, Hernandez-Diaz AG, Gortazar F, Hinojosa MA (2018) A multiobjective GRASP/VND algorithm to solve the waste collection problem. Int Trans Oper Res 25:545–567

Moon I, Nguyen TVL (2014) Container packing with balance constraints. OR Spectr 36:837–878

Morabito R, Morales S (1998) A simple and effective recursive procedure for the manufacturer's pallet loading problem. J Oper Res Soc 49(8):819–828

Morabito R, Morales S, Widmer J (2000) Loading optimization of palletized products on trucks. Transp Res Part E Logist Transp Rev 36(4):285–296

Moura A, Bortfeldt A (2017) A two-stage packing problem procedure. Int Trans Oper Res 24:43–58

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

ORTEC (2018) Company. www.ortec.com. Accessed 04 Mar 2018

Parreño F, Alvarez-Valdes R, Oliveira JF, Tamarit JM (2010) A hybrid GRASP/VND algorithm for two- and three-dimensional bin packing. Ann Oper Res 179:203–220

Peng B, Liu M, Lu Z, Kochengber G, Wang H (2016) An ejection chain approach for the quadratic multiple knapsack problem. Eur J Oper Res 253:328–336

Pisinger D (2000) A minimal algorithm for the bounded knapsack problem. INFORMS J Comput 12(1):75–82

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

Queiroz T, Miyazawa F (2013) Two-dimensional strip packing problem with load balancing, load bearing and multi-drop constraints. Int J Prod Econ 145:511–530

Queiroz T, Miyazawa F (2014) Order and static stability into the strip packing problem. Ann Oper Res 223:137–154

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, Gonçalves JF, Lopes MP (2016) A container loading algorithm with static mechanical equilibrium stability constraints. Transp Res Part B Methodol 91:565–581

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

Resende MGC, Werneck RF (2004) A hybrid heuristic for the p-median problem. J Heuristics 10:59–88

Sheng L, Hongxia Z, Xisong D, Changjian C (2016) A heuristic algorithm for container loading of pallets with infill boxes. Eur J Oper Res 252:728–736

Takahara S (2005) Loading problem in multiple containers and pallets using strategic search method. In: Torra V, Narukawa Y, Miyamoto S (eds) Modeling decisions for artificial intelligence, vol 3558. Lecture notes in computer science. Springer, Berlin, pp 448–456

Toffolo T, Esprit E, Wauters T, Vanden Berghe G (2018) A two-dimensional heuristic decomposition approach to a three-dimensional multiple container loading problem. Eur J Oper Res 257(2):526–538

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

Zachariadis EE, Tarantilis CD, Kiranoudis CT (2012) The pallet-packing vehicle routing problem. Transp Sci 46:341–358

Zhao X, Bennell J, Betkas T, Dowsland K (2016) A comparative review of 3D container loading algorithms. Int Trans Oper Res 23:287–320