Research Article

Algorithms for Pallet Building and Truck Loading in an Interdepot Transportation Problem

**Maria Teresa Alonso,¹ Ramon Alvarez-Valdes,²
Francisco Parreño,¹ and Jose Manuel Tamarit²**

¹*Department of Mathematics, University of Castilla-La Mancha, Avenida de España, 02071 Albacete, Spain*

²*Department of Statistics and Operations Research, University of Valencia, Doctor Moliner 50, Burjassot, 46100 Valencia, Spain*

Correspondence should be addressed to Ramon Alvarez-Valdes; ramon.alvarez@uv.es

Received 15 December 2015; Accepted 15 March 2016

Academic Editor: Paolo Maria Mariano

Copyright © 2016 Maria Teresa Alonso et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

This paper deals with the problem of a logistics company that has to serve its customers by first putting the products on pallets and then loading the pallets into trucks. Besides the standard geometric constraints of products not overlapping each other and not exceeding the dimensions of pallets and trucks, in this real problem, there are many other constraints, related to the total weight of the load, the maximum weight supported by each axle, and the distribution of the load inside the truck. Although the problem can be decomposed into two phases, pallet loading and truck loading, we have taken a combined approach, building and placing pallets at the same time. For each position in the truck, a pallet is built and tailored for that position according to the constraints of height and weight. We have developed a GRASP algorithm, in which the constructive algorithm is randomized and an improvement phase is added to obtain high-quality solutions. The algorithm has been tested on two sets of real instances with different characteristics, involving up to 44 trucks. The results show that solutions with an optimal or near optimal number of trucks are obtained in very short computing times.

1. Introduction

A distribution company has a main depot and a set of regional depots spread around the country and every week it has to solve an interdepot transportation problem to fill the orders that each regional depot sends to the main depot for the days of the next week. The company has to decide first how to put the products into pallets and then how to distribute the pallets among trucks in order to deliver the right products to the right place at the right time, while minimizing the number of trucks used. In this paper, we will take the characteristics and the data of a large distribution company as a reference, but the problem is common to many other logistics companies around the world. The companies require a solution procedure that produces high-quality solutions in short computing times, to be able to react to last-minute changes. Good solutions, minimizing the number of trucks, will result in reductions in the transportation costs for the companies as well as in greenhouse gas emissions.

The loading problem faced by the company consists of two interrelated phases. In the pallet building phase, all the products have to be put on pallets. In the truck loading phase, all the pallets have to be placed into trucks. The means of transport, in this case trucks, introduces some constraints which have to be respected for safety reasons. There is a maximum weight of the load that cannot be exceeded. There are also limits on the maximum weight each axle can support. Moreover, the load has to be well distributed in the truck to prevent movements during the journey. This means that the center of gravity of the load always has to be placed between the axles and as near as possible to the geometric center of the truck.

In order to produce fast and high-quality solutions, we have developed a GRASP algorithm, composed of a constructive algorithm, designed to satisfy all the safety and logistic constraints of the problem, a randomization procedure to achieve diversity in the iterative process, and some improvement moves to refine the solutions obtained

by the randomized constructive algorithm. A novel feature of the proposed algorithm is that it solves the two phases of the problem simultaneously, instead of decomposing it into two separate subproblems.

Although the algorithm provides solutions for the orders of all the days in the planning horizon, the company sends the trucks with the products for the first day and adjusts its data for the next days taking into account last-minute orders or cancelations, solving the problem again for the remaining days. Nevertheless, solving the whole problem together is much better than solving each day separately, because 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, because products can be sent before their delivery date but not later. By solving the whole problem, the trucks are more efficiently used.

The remainder of the paper is organized as follows. An overview of the related existing work is presented in Section 2. In Section 3, the problem is formally described and all the constraints related to the pallet building and the truck loading are discussed. In Section 4, the different components of the GRASP algorithm are introduced. Section 5 contains the description of the test instances used, which come from a large distribution company, and the results obtained by the algorithm. The comparison with simple lower bounds developed for the problem shows that the proposed procedure obtains high-quality solutions for medium and large instances, involving up to 44 trucks. Section 6 contains the conclusions.

2. Literature Review

There are not many papers addressing the issues studied here, considering pallet and truck loading together and multiple trucks. Morabito et al. [1] deal with the same problem but in two dimensions, because the products cannot be stacked. Both problems, pallet and truck loading, are solved by using the 5-block algorithm proposed by Morabito and Morales [2]. Takahara [3] deals with the problem of loading a set of items on a set of containers and pallets. The sequences for loading the items and for selecting the bins are controlled by heuristic procedures. Haessler and Talbot [4] propose a heuristic for developing load patterns for trucks and rail shipments. The products have low density and the approach is based on loading by volume rather than by weight. To deal with axle weight constraints, stacks are sequenced by alternating the heaviest and lightest stacks. Moura and Bortfeldt [5] deal with the same problem in two steps. In the first step, boxes are packed onto pallets, while in the second step these pallets are loaded into trucks. For packing boxes onto pallets, Moura and Bortfeldt use the algorithm proposed by Moura and Oliveira [6] and they deal with the problem of loading pallets into trucks as a one-dimensional bin packing problem, which is solved by a tree search procedure. Zachariadis et al. [7] address a vehicle routing problem in which the products are first packed onto pallets and then loaded into the vehicles and propose local search metaheuristic strategies. Alonso et al. [8] consider a simplified version of the problem studied here and develop integer linear models, starting from a basic loading

model and progressively adding more realistic conditions and objectives, such as the use of one or two pallets per position, the position of the center of gravity, and the minimization of the number of pallets.

The Single Container Loading Problem (SCLP), in contrast, is a well-studied problem and in recent years there has been an increasing number of papers introducing real constraints (see the survey by Bortfeldt and Wäscher [9]). Gehring and Bortfeldt [10], Bortfeldt et al. [11], Terno et al. [12], and Egeblad et al. [13] are some of the authors who include weight limit constraints in their studies. In fact, when the cargo is heavy, weight becomes a very restrictive constraint, more than the volume or the space occupied.

Weight distribution constraints require the weight of the cargo to be distributed across the container floor to balance the load. To achieve a good weight distribution, the center of gravity of the load should be at the geometrical midpoint of the container floor, as in Bischoff and Marriott [14], or at a point not exceeding a certain distance from it, as in Bortfeldt and Gehring [15] and Gehring and Bortfeldt [10].

Axle weight is a constraint imposed by the means of transport and it has not been widely studied. Lim et al. [16] deal with a particular SCLP with axle weight constraints. They first apply a customized wall-building heuristic based on the GRASP by Moura and Oliveira [6], including special considerations for box weight and density. They then use an integrated approach to handle the weight requirements. If the container load limit is exceeded, they unload the necessary number of boxes by iteratively solving an ILP model to meet the requirement. If the axle weight limit is exceeded, they take two steps iteratively until the limit is satisfied: the first step consists in reversing the positions of the walls created by the customized heuristic, whereas the second step involves solving an ILP model to unload boxes and applying the first step once more to improve the balance of the container as well as to force a feasible weight distribution.

Stackability or load-bearing constraints are introduced to avoid damaging the items at the bottom of the stacks. They can be measured by the number of boxes a box can bear above it (Bischoff and Ratcliff [17]) or by prohibiting a particular type of box being placed on top of another type (Terno et al. [18]) or by the maximum weight that can be applied to a box per unit area (Junqueira et al. [19], Alonso et al. [20]).

Other constraints are related to the stability of the load. Vertical or static stability prevents items from falling when the vehicle is not moving (Ramos et al. [21]). A box must be supported from below at a given percentage of the surface of its base. If this percentage is 100%, we speak of full support (De Araújo and Armentano [22], Fanslau and Bortfeldt [23]). In the case of lower percentages, we speak of partial support (Junqueira et al. [19]). Horizontal or dynamic stability assures that items do not move while the container is being moved. This constraint deals with the capacity of the items to support the inertia of their bodies (Ramos et al. [24]).

Weight constraints, cargo stability, and other practical constraints also appear in recent studies combining loading and routing, such as Iori et al. [25], Bortfeldt [26], Junqueira et al. [27], Ceschia et al. [28], Tao and Wang [29], and Junqueira and Morabito [30]. Doerner et al. [31] deal with a particular

vehicle routing problem in which the items are placed on pallets and stacked one on top of the other, producing piles. They propose two metaheuristic algorithms, a Tabu Search and an ACO algorithm. Pollaris et al. [32] combine a capacitated vehicle routing problem with the loading of homogeneous pallets inside the vehicle. They consider sequence-based pallet loading and axle weight constraints and propose a mixed ILP formulation for the problem. Their model tries to minimize the transportation cost, respecting the axle weight constraint along the delivery route. Recent surveys on loading and routing problems have been presented by Iori and Martello [33, 34] and Pollaris et al. [35].

3. Description of the Problem

The problem begins when a depot sends a set of orders for the upcoming days $d \in 1, \dots, D$. For each day d , the order consists of a list of products j , $j \in 1, \dots, J$, with their required number of items for this day, a_{jd} . The products have to be served that day, or before, but not later. Each product is defined by its dimensions (x_j, y_j, z_j) , its weight w_j , and its possible rotations (rx_j, ry_j, rz_j) , where $rx_j = 1$ if the dimension x_j of the product can be upward.

All the products have a predefined placement in layers. A layer is an arrangement of items of the same product j in rows and columns, forming a rectangle defined by its dimensions (l_j, w_j, h_j) , its weight q_j , and the number of items in the layer lu_j . The composition of the layers is previously determined by the company. For each type of product, the company has defined the items by row and by column, their orientation, and their placement. A layer covers the pallet base in horizontal directions, guaranteeing sufficient support for other layers that can be placed on top of it to make up the pallet.

The quantity of layers per product and day is $n_{dj} = \lfloor a_{jd}/lu_j \rfloor$, whereas the total quantity of layers of a product is $n_j = \sum_{d \in D} n_{dj}$. The demand for each type of product is a number of layers. If there are some remaining items of a product not included in a layer, these items will form a special pallet.

The loading problem consists of two interrelated phases:

- (i) *Pallet Building Phase*. All layers have to be placed on pallets.
- (ii) *Truck Loading Phase*. All pallets have to be placed in one of the trucks.

3.1. Pallet Building. In this phase, we have to place the layers on pallets. A pallet has a base, made of wood, metal, or plastic, with dimensions (l^p, w^p, h^p) and weight q^p , and it can be rotated horizontally. A pallet provides a loading space above its base. Typically, the space in horizontal directions is limited by the horizontal dimensions of the pallet, though some overhang in both directions can be allowed. In the vertical direction, the height of the loaded pallet is limited to half the height of the truck, so that two pallets can be placed one on top of the other.

A pallet is thus composed of a flat base and a set of layers placed one on top of the other. Two types of pallets

can be built, *stock pallets*, where all the layers are of the same product, and *case pallets*, formed by layers of different products. The company wants to make as many stock pallets as possible. Only when not all the layers of a product fit onto stock pallets are the remaining layers used to form case pallets. A third type of pallet, the *rest pallet*, appears when the demand for some product, a_{jd} , is not a multiple of lu_j and there are some remaining items not included in any layer. All these remaining items are put together on one or more rest pallets.

Apart from pallet composition, other constraints should be considered in this phase.

(i) *Orientation Constraint*. Layers can be rotated, but all placements must be parallel to the sides of the pallet.

(ii) *Support Constraint*. A certain minimum percentage of a layer, 75% to be precise, must be supported by the pallet base or by the layer on which it rests.

(iii) *Priority Constraint*. A pallet can contain only items of orders for the same day.

(iv) *Stackability Constraint*. The number of layers that can be stacked is limited to avoid product damage. It depends on their weight, with lighter layers being placed on top of heavier ones, and on the fragility of the products. For each type of product, the maximum number of layers that can be stacked, ls_j , is defined by the company, and the stock pallets are composed of this number of layers. However, when we build case pallets, with different types of layers in the same pallet, we have to take their stackability into account. Each product belongs to a stackability group which is identified by a value $ids_j \in 1, \dots, SG$, linked to the density of the product, with the densest products having an identification of 1 and the least dense having the maximum value SG . One layer can be placed on top of another if the product in the layer below has a stackability identification smaller than or equal to that of the product in the layer above.

3.2. Truck Loading. Once the pallets are prepared, they have to be placed in the trucks. We have a set of trucks $k \in 1, \dots, K$, all of the same type, with two axles and with unlimited availability. More precisely, for each truck, we know its dimensions (L, W, H) and the maximum weight, or *weight capacity* Q , that it is allowed to load. Moreover, there are restrictions on the maximum weights the axles can support. The truck has two axles which carry the load, with the front axle at position δ_1 and the rear axle at position δ_2 , where δ_1 and δ_2 are measured from the front of the loading space of the truck. Figure 1 shows a truck with its dimensions and axle distances.

For security reasons, the load has to be well spread out over the truck and must be distributed between the two axles, not exceeding the maximum weight they can bear. If Cog_x is the x -coordinate of the center of gravity of the load, $\text{Cog}_x = \sum_p x_p W_p / g$, where x_p is the position of a pallet and W_p is its weight and g is the total weight of the load; the weights

FIGURE 1: A truck with the positions of the axes.

supported by the front axle (g_1) and the rear axle (g_2) are determined by

$$\begin{aligned} g_1 &= \frac{g(\text{Cog}_x - \delta_1)}{\delta_2 - \delta_1}, \\ g_2 &= \frac{g(\delta_2 - \text{Cog}_x)}{\delta_2 - \delta_1}. \end{aligned} \quad (1)$$

Each axle can bear a maximum weight, denoted by Q_1 and Q_2 . Usually, $Q_1 + Q_2 \geq Q$. Therefore, we impose the conditions $g_1 \leq Q_1$ and $g_2 \leq Q_2$, as well as $\delta_1 \leq \text{Cog}_x \leq \delta_2$, ensuring that the center of gravity is always between the axes.

The expressions of the weight supported by the axes, provided by the company, are based on the total weight being put on the truck and the center of gravity of the load. Although they are nonlinear, they can be easily computed in the constructive process developed in this study. An alternative linear formulation, based on the contribution of each item to the weight supported by the axes, has been used in [8], with the information contained in [36].

In this phase of truck loading, some other constraints related to how to place the pallets should be considered.

(i) *Priority Constraint.* The delivery days of the orders must be respected, meaning that each day the trucks being sent to the depot must contain all the products required for that day, although some of them may have been sent before. Each day d , only trucks containing products for that day are sent. Trucks in the solution that are completely filled with products for days $d + 1, d + 2, \dots$ are not sent on day d . However, if the items for day d do not completely fill a truck, the items for day $d + 1$ can be used to fill up the truck. If after adding these items the truck is not still completely filled, then items for day $d + 2$ can be used and so on, until the truck is filled or there are no items left.

This strategy could result in some products being sent several days before their due date in some cases. However, no

constraints or penalties are considered to limit this situation. On the one hand, in practice, it affects very few products for more than one day or two. On the other hand, the company gives higher priority to minimizing the number of trucks.

(ii) *Stability Constraint.* For stability reasons, the pallets have to be placed touching at least one side of the truck.

(iii) *Stackability Constraint.* The pallet height is limited to half the truck height. Therefore, at most two pallets can be placed in each stack. However, one cannot place any two pallets in the same stack; it depends on their weight, with lighter pallets being placed on top of heavier pallets. Each pallet belongs to a stackability group identified by a value $\text{ids}_j \in 1, \dots, \text{SG}$. The stackability identification of a pallet ids_j is the minimum stackability identification of the products that compose the pallet. One pallet can be placed on top of another if the pallet below has a stackability identification smaller than or equal to that of the pallet above.

3.3. *Objective.* When preparing the transportation plan for the next few days, the available data are usually not complete, except for the next day: orders may be canceled or, more probably, increased with new products for the upcoming days. Consequently, the distribution company will not ship the goods for all the days, but only the first trucks, containing the next day's orders and possibly some orders for the following day(s).

Nevertheless, the company wants the problem solved for all the orders of all the days in the planning horizon, because the orders for the later days might influence the choices made for the first days. For this reason, our main objective is to minimize the total number of trucks needed for all the orders, and for solutions with the same number of trucks, one is better than the other if the last truck is less full, leaving more room for new orders. The inclusion of this secondary objective of leaving the last truck as empty

as possible for future use makes the problem similar to cutting stock problems with usable leftovers, which have been extensively studied in one and two dimensions due to their many practical applications (Arenales et al. [37], Andrade et al. [38]).

We are tackling a problem with two parts to be solved simultaneously: putting products onto pallets and pallets into trucks, but they have similar characteristics. The type of assignment is input minimization, whereby a set of small items is to be assigned to a set of large objects whose quantity is large enough to accommodate all the small items. The assortment of the small items is weakly heterogeneous; the small items can be grouped in relatively few classes compared with the total number of items. The assortment of the large objects is that there is only one large object type with fixed dimensions, pallet and truck, and the shape of the small items is regular. According to the typology for cutting and packing problems introduced by Wäscher et al. [39], the truck loading part can be classified as a Three-Dimensional Single Stock Size Cutting Stock Problem (3D-SSSCSP), while the pallet building part is one-dimensional (1D-SSSCSP), except for the rest pallet problem, which is three-dimensional.

4. Approach to the Problem

We can consider the problem as two subproblems. The first subproblem is building the pallets, where the input is the list of all the orders for all the days in the planning horizon and the output is a pallet list with all the products to send. The second problem is how to load the pallets onto the trucks, where the input is the list of pallets provided by the first subproblem and the output is the number of trucks needed to load the whole list of pallets from the first subproblem.

The disadvantage of this approach is that the two subproblems are independent and one does not take into account the features and constraints of the other. In particular, in the first subproblem, the pallets are built without any information about the trucks. Therefore, a good solution for the pallet loading phase is not necessarily a good starting point for the truck loading phase.

Our proposal is to keep a constant flow of information between the two subproblems, taking into account the constraints and the features of both. In the truck loading phase, we gather information about the features of the pallet, whereas, in the pallet building phase, we collect information about the placement of the pallet in the truck. With both sets of information, we build the best pallet for a specific position in the truck using the remaining products.

To solve this problem, we propose a GRASP algorithm with a constructive phase in which a solution is built by adding a pallet at each step according to the information collected from the truck, a randomization strategy to obtain diverse solutions in an iterative process, and an improvement phase. In the following sections, the elements of the algorithm are described in detail.

4.1. Constructive Phase. The constructive algorithm builds a solution by means of an iterative process in two steps, the

first of which consists of finding a space in the truck to place the pallet and the second consists of building the pallet and placing it into the truck.

Step 1 (selecting a place). The first step is to select a position in the truck to place the next pallet. If there were no oversize layers forming oversize pallets, each pallet would have a predefined position in the truck because all the pallets would have the same dimensions. But if there are some oversize pallets in which some layers are larger than the base, the position of the pallet cannot be predefined, because it depends on the pallet dimensions. That is why we have to select a place to put the pallet at each step.

Other constraints that have to be taken into account when selecting the place are the center of gravity and the weight on each axle, because the position of the pallet affects the weight that each axle will bear and may move the center of gravity away from the geometric center of the truck.

Since the weight has to be balanced between the two axles, we have decided to divide the truck into two parts, front and rear, starting from the center. By placing pallets alternately in each part, we can control the balance of the weight and keep the center of gravity approximately in the center of the truck.

The center of gravity and the weight each axle will bear are calculated by the expressions described in Section 3.2. If the weight that axle 1 can still bear, $Q_1 - g_1$, is greater than that of axle 2, $Q_2 - g_2$, the next space is selected at the front; otherwise, it is chosen at the back.

Starting at the center of the truck and going in the chosen direction, towards the front or the back, we look for a space big enough to place a pallet. We search the floor of the truck first and if there is no space left, we look at the top of the pallets already loaded. When the space is found, its features are extracted: its dimensions (S_x, S_y, S_z), the maximum weight that can be put into this position W_s , and the stackability identification of the lower pallet, ids_s , if the space is above another pallet.

Having determined these features of the chosen space, we move on to the next step.

Step 2 (building a pallet for the selected space). In this step, we build a pallet that is tailored to the selected space. When building a pallet, we always use the predefined layers, except for the special case of rest pallets.

The layers are ordered by day, first days first, and by density, the denser ones first. We take the first layer on the list and check whether there are enough layers of this product to build a stock pallet. If this is the case, we build it. Otherwise, these layers are added to the pallet and we continue searching for layers that cannot form a stock pallet by themselves and add them to the pallet, building a case pallet with layers of different products.

A layer is selected if it satisfies several conditions. First, it fits into the selected space and is supported in at least 75% of its area by the layer below it. Second, the weight of the pallet, including this layer, does not exceed the maximum weight allowed for this space and keeps the center of gravity between the axles. Third, the height of the pallet does not exceed half the height of the truck if it is placed on the floor. If it is placed

FIGURE 2: Arrangement of the first level.

FIGURE 3: Pallets arranged on the second level.

above another pallet, the sum of the pallets heights cannot exceed the truck height.

The pallet built in this way is defined by its dimension, (l^p, w^p, h^p) , the number of layers nl^p , the total weight q^p , and the stackability identification, which is the maximum identification of all the layers making up the pallet.

The pallet is placed in the selected space, touching the nearest wall of the truck, in such a way that the gap, if any, is left in the middle of the truck, where it can be used by pallets on the other side and also by pallets on top, always satisfying a 75% area of support, as can be seen in Figures 2 and 3.

When the pallet is inserted into the solution, a new cycle begins, calculating the new center of gravity, the load supported by the axles, and the direction to be chosen and updating the list of layers.

When the first level on the floor of the truck is complete, all the pallets are pushed to the front of the truck. The

placement of the pallets, starting from the center, may leave two gaps, one at the front and another at the back of the truck, as can be seen in Figure 2. By pushing the pallets to the front of the truck, we merge the gaps at the end, with the possibility of placing another pallet in that gap. This process is performed before starting the placement on the second level, but it takes place only if the modified location of the pallets is feasible. If it is not, the containers are kept where they are and we go to the second level. In the many instances studied, this movement of the pallets on the first level has nearly always been possible. In the very few cases in which it was not, it was due to the presence of very heavy pallets, almost reaching the total weight limit or the limits on the axles. In these cases, moving the existing pallets would not have allowed the placement of another pallet.

The constructive phase continues until all the layers have been placed or until no more pallets can be placed in the truck because one of the constraints, limiting the volume, the weight, or the number of pallets, has been reached. If there are still pallets to be sent, a new truck is opened and the process continues.

4.2. Randomized Constructive Algorithm. In the construction phase, the solution is built step by step by adding one element to a partial solution. In order to select the element to add, a greedy randomized strategy, *Sample plus Construction*, is used to add diversity to the process. When the space is selected, we consider all the layers that satisfy all the conditions for being placed in that position and take a random sample of these layers. The number of layers on the list is controlled by a parameter γ ($0 \leq \gamma \leq 1$), as proposed by Resende and Werneck [40]. Each layer in the sample is evaluated and the densest layer is chosen. Then, we follow the process explained previously to complete the pallet. The diversity of the process is managed by the value of parameter γ . If γ is equal to 1, all the layers are selected for the sample and the densest of all the layers is always chosen, making the selection deterministic. If, on the contrary, γ is near 0, the number of selected candidates is smaller, the densest layers have less probability of being selected, and another less good candidate could be chosen, making the selection more random.

It is difficult to determine the value of γ , the fraction of elements composing the list, that gives the best overall results. We follow a reactive greedy adaptive search procedure, proposed by Prais and Ribeiro [41], in which the value of the parameter γ is self-tuned according to the quality of the results. The parameter γ is initially taken at random from a set of discrete values $\Gamma = \{0.1, 0.2, \dots, 0.9\}$, and at each iteration, γ takes a random value from this set. After a certain number of iterations, 500 in the current implementation, the relative quality of the solutions obtained with each value of γ is evaluated. The probability of the selection of each value is recalculated according to the quality of the solutions, increasing the probability of the values producing better solutions and decreasing the others. In this way, the randomization procedure of GRASP reacts to the results

obtained in the iterative process, tuning the parameter to the most suitable values for each instance.

4.3. Improvement Phase. A solution of the randomized constructive phase consists of a list of pallets packed on each truck, with their contents and positions, and some characteristics of each truck, in particular the weight on each axle. There are two reasons for closing a truck; one is that the maximum weight an axle can support is reached, because the pallets are heavy, and the other is because all the positions are occupied. One solution can combine trucks closed by weight and others closed by positions.

In order to improve the solutions, three movements have been developed, taking into account the reasons for closing a truck, and they are applied according to the features of the solution, trying to improve it. In the next subsections, the movements are described.

Nevertheless, not all the solutions go into the improvement phase. We only consider promising ones, that is, those which are below a certain threshold. At the beginning, the threshold takes the value of the first solution of the constructive algorithm. Then, if at an iteration the solution value is lower than the threshold, we update this threshold and go to the improvement phase. If the solution value is greater than the threshold, the solution is rejected and the reject counter is increased. When the number of rejected solutions is greater than a value maxFilter , the threshold is increased according to the expression: $\text{threshold} = \text{threshold} + \lambda(1 + \text{threshold})$, where λ is set at 0.2 and $\text{maxFilter} = 5$, as in Marti and Moreno-Vega [42].

4.3.1. Improvement 1: Filling a Percentage of the Trucks. The first improvement move consists of eliminating a percentage of the last trucks in the solution and refilling these trucks using another strategy.

In our implementation, a percentage of the trucks, selected at random from the interval (50%, 80%), is eliminated. The partial solution is then completed using three different strategies:

(i) *Deterministic by Volume.* The list of the remaining layers is ordered by day and by volume, and the deterministic constructive algorithm is applied.

(ii) *Randomized by Volume.* The randomized constructive algorithm, described in Section 4.2, is applied to fill the truck again, but ordering the products first by day and then by volume, instead of ordering them by density.

(iii) *Deterministic by Weight.* The list of the remaining layers is ordered by day and by weight, and the deterministic constructive algorithm is applied.

The best solution for the next truck provided by the three strategies is added to the solution and the process continues with the next truck, until all the products are packed. At the end of this process, if the solution obtained is better than the initial one, we keep it. Otherwise, we keep the initial solution.

This move is applied to all trucks, regardless of the closing condition.

4.3.2. Improvement 2: Swapping and Insertion. This improvement move is only for those trucks that have been closed because an axle has reached its maximum weight; we try to obtain better results by swapping pallets between consecutive trucks. The idea is to move heavy pallets from the first truck A to the next truck B so that the axles of truck A carry less weight, allowing us to load more pallets. In this way, we try to put as many pallets as possible in the first trucks, extracting them from the following ones. Consequently, if we are able to empty the final truck, we save a truck in the complete solution.

This swapping movement must be followed by an insertion in order to increase the number of pallets in truck A. First, each pallet in truck A and all the pallets in truck B are examined in order to find candidates for swapping. Two pallets are swapped if the pallet in truck B can fit into the place of the pallet in truck A and it is lighter. Then, we try to take advantage of the difference in weight by loading more pallets. If we find candidates for swapping, we evaluate the remaining pallets in truck B to check whether one of them can be inserted into a gap in truck A. We only consider this improvement if we find successful candidates for both movements, swapping and insertion.

As a consequence of this movement, some empty spaces can appear inside truck B, because two pallets are moved from B to A and only one pallet is moved from A to B. The next movement in B consists of inserting pallets from the next truck C, refilling the empty spaces. The algorithm looks for candidates among the pallets of C that can fit into this gap in B in such a way that when the weight of the candidate is added to the weight in B, the total weight does not exceed the weight capacity of the truck and the maximum weight that can be supported by the axles. Once these previous restrictions are satisfied, the candidate can be placed into the empty space in truck B. If this empty space is on the second level of the truck, the pallet is put there if its stackability identification is larger than or equal to that of the pallet below. Otherwise, the new pallet is put on the first level and the other pallet is put on top of it.

The idea is to try to move pallets from the later trucks to the previous ones, in order to reduce the number of pallets in the last truck as much as possible, even reducing the solution by one truck, if all the pallets in the last truck are moved to other trucks.

A pseudocode of the complete algorithm appears in Algorithm 1. Since two objectives are considered, the value of solution is the total volume used, $V = L * W * H * (\#trucks - 1) + \text{volume of the pallets in the last truck}$.

5. Computational Results

The algorithm was coded in C++ and run on an Intel Core i7-4790, 3.60 GHz with 16 GB of RAM. In this section, we describe first the instances used, then the lower bounds developed to assess the performance of the algorithm, and finally the results obtained.

Require: $\Gamma = \{0.1, 0.2, \dots, 0.9\}$, set of values for γ

- (1) $V_{\text{best}} = \infty$; $V_{\text{worst}} = 0$; $V = 0$
- (2) $n_{\Gamma_{\gamma}^*} = 0$, number of iterations with each γ value
- (3) $\text{Sum}_{\Gamma_{\gamma}^*} = 0$, sum of the volumes of the solutions obtained using γ
- (4) $P(\Gamma_{\gamma} = \Gamma_{\gamma}^*) = P_{\Gamma_{\gamma}^*} = 1/|\Gamma|$, $\forall \Gamma_{\gamma}^* \in \Gamma$, probability of using value γ
- (5) $\text{numIter} = 0$, $\text{niter} = 0$
- (6) $Q = \infty$, quality threshold
- (7) **while** $\text{numIter} < \text{maxIter}$ **do**
- (8) Choose a γ value, $\Gamma_{\gamma} = \Gamma_{\gamma}^*$, with probability $P_{\Gamma_{\gamma}^*}$
- (9) $n_{\Gamma_{\gamma}^*} = n_{\Gamma_{\gamma}^*} + 1$
- (10) $\text{numIter} = \text{numIter} + 1$
- (11) $V = \text{Randomized_Constructive_Algorithm}(\gamma)$
- (12) **if** $V < Q$ **then**
- (13) $Q = V$
- (14) $V = \text{Improvement_Phase}(V)$
- (15) **else**
- (16) $\text{niter}++$
- (17) **if** $\text{niter} > \text{maxFilter}$ **then**
- (18) $Q = Q + 0.2(1 + Q)$
- (19) $\text{niter} = 0$
- (20) **end if**
- (21) **end if**
- (22) **if** $V < V_{\text{best}}$ **then** $V_{\text{best}} = V$ **end if**
- (23) **if** $V > V_{\text{worst}}$ **then** $V_{\text{worst}} = V$ **end if**
- (24) $\text{Sum}_{\Gamma_{\gamma}^*} = \text{Sum}_{\Gamma_{\gamma}^*} + V$
- (25) $\text{mean}_{\Gamma_{\gamma}^*} = \text{Sum}_{\Gamma_{\gamma}^*} / n_{\Gamma_{\gamma}^*}$
- (26) **if** $\text{mod}(\text{numIter}, 500) == 0$ **then**
- (27)
$$\text{eval}_{\Gamma_{\gamma}^*} = \left(\frac{V_{\text{worst}} - \text{mean}_{\Gamma_{\gamma}^*}}{V_{\text{worst}} - V_{\text{best}}} \right)^{\alpha}, \quad \forall \Gamma_{\gamma}^* \in \Gamma$$
- (28)
$$P(\Gamma_{\gamma}^*) = \frac{\text{eval}_{\Gamma_{\gamma}^*}}{\sum_{\Gamma_{\gamma'} \in D} (\text{eval}_{\Gamma_{\gamma'}})}, \quad \forall \Gamma_{\gamma}^* \in \Gamma$$
- (29) **end if**
- (30) **end while**

ALGORITHM 1: ReactiveGRASP.

5.1. Description of the Datasets. We have two sets of test instances, with different characteristics. They are real instances derived from the everyday transportation activity of a large distribution company. The instances have been provided to us by ORTEC [43], a company that develops planning and optimization solutions and services for logistics companies.

(i) *Set I, Comprising 77 Instances.* The orders correspond to three upcoming days. Only stock pallets and case pallets are considered, because the demand for the products is always given as an integer number of layers. Each product has a stackability identification, ranging from 1 to 4. All the instances use one type of pallet, the ISO pallet (1016 mm, 1209 mm), and one type of truck, with two axles.

(ii) *Set II, Comprising 111 Instances.* The orders correspond to only one day. Only case pallets are considered. There are no stackability identifications and therefore any pallet (layer) can

be put on top of any other pallet (layer). The dimensions of the pallets and trucks vary among the instances, but each instance uses one type of pallet and one type of truck, with two axles.

5.2. Lower Bound. We can calculate some simple lower bounds on the number of trucks required for each instance, according to the weight, the volume, and the number of positions in a truck. The bound based on the weight is the sum of the weights of all the required products divided by the weight capacity of the truck. The bound based on the volume is calculated in the same way, using the volume of the products. If we define the maximum number of positions in a truck, $I = \lfloor (L / \min\{l^p, w^p\}) * 4 \rfloor$, that is, the maximum number of pallets that can be put along the length of the truck multiplied by 4 (two rows, two levels), then the bound based on the number of positions is calculated by dividing the sum of the heights of the layers by the truck height and by the number of positions in a truck. The maximum of these

TABLE 1: Results of the deterministic constructive algorithm.

	Sum of LB	Trucks	Pallets	Av. time
Set I	511	581	23525	0.04
Set II	1113	1194	38367	0.03

TABLE 2: Distance to the bound of the solutions of the deterministic constructive algorithm.

	Distance to bound			
	0	1	2	3
Set I	23	38	16	0
Set II	41	60	9	1

three bounds, L_{init} , given by (2), is a valid lower bound on the number of trucks required for each instance:

$$L_{\text{init}} = \max \left\{ \left\lceil \frac{\sum_{j \in J} q_j * n_j}{Q} \right\rceil, \left\lceil \frac{\sum_{j \in J} l_j * w_j * h_j * n_j}{L * W * H} \right\rceil, \left\lceil \frac{\sum_{j \in J} h_j * n_j}{H * I} \right\rceil \right\}. \quad (2)$$

5.3. Computational Results. Table 1 shows the results of the constructive deterministic algorithm. For each set of instances, the table contains the sum of the lower bounds on the required trucks, the total number of trucks used, the total number of pallets built, and the average running time per instance in seconds. The sum of the lower bounds for the instances in Set I is 511 and for the instances in Set II it is 1113. Therefore, the sum of the solutions exceeds the sum of the lower bounds by 70 trucks for Set I and by 81 trucks for Set II, less than one truck per instance on average. Table 2 shows the distribution of the distance from the solution to the lower bound for each instance.

The distances observed in Table 2 are due to two factors. On the one hand, the lower bound does not take into account some of the characteristics of the problem, such as the structure of the demand by days, the stackability conditions, and the limits on the weight supported by the axles. On the other hand, the constructive algorithm uses a simple criterion, based on the product density, to select the layers composing the pallets. For instances with heavy products, this criterion may produce a solution in which the first truck is closed because some limit on the weight is reached, but there is plenty of unused space, while in other later trucks the weight limits are not reached but the trucks are closed because all the positions are used. The randomization of the constructive algorithm and the improvement moves have been designed to correct these undesirable situations.

In Table 3, we can see the results of the randomized constructive algorithm, with a stopping criterion of 1500 iterations. The algorithm has also been run for larger numbers of iterations (3000, 5000), but the improvements are small and they are not worth the increase in running times.

For Set I, we observe an overall reduction of 19 trucks. The reduction is smaller for Set II, only 8 trucks. Table 4

TABLE 3: Results of the randomized constructive algorithm.

		Trucks	Pallets	Av. time
Set I	Deterministic	581	23525	0.04
	Randomized	562	24446	57
Set II	Deterministic	1194	38367	0.03
	Randomized	1186	38960	42

TABLE 4: Distance to the bound of the solutions of the randomized constructive algorithm.

		Distance to bound				Pallets in last truck
		0	1	2	3	
Set I	Deterministic	23	38	16	0	1328
	Randomized	30	43	4	0	1069
Set II	Deterministic	41	60	9	1	1993
	Randomized	46	58	6	1	1784

TABLE 5: Results of the GRASP algorithm.

		Trucks	Pallets	Av. time	Pallets in last truck
Set I	Deterministic	581	23525	0.04	1328
	Randomized	562	24446	57	1069
	GRASP	562	24449	127	812
Set II	Deterministic	1194	38367	0.03	1993
	Randomized	1186	38960	42	1784
	GRASP	1186	38936	116	1774

completes the comparison between the deterministic and the randomized versions of the constructive algorithm. As well as the distances to the bounds for the two algorithms, the last column shows the number of pallets in the last truck for the instances in which both algorithms obtain the same number of trucks. As stated in Section 3.3, for solutions with the same number of trucks, a solution with fewer pallets in the last truck is preferable, because it has more empty space for accommodating last-minute orders. If the number of pallets in the last truck is very low, the planner may decide to use a smaller vehicle or even not to send it and wait for future orders to fill the truck. Table 4 shows that the overall reduction is 259 pallets for the instances in Set I and 209 pallets for the instances in Set II.

The results obtained by the GRASP algorithm, when the improvement phase described in Section 4.3 is added to the randomized constructive algorithm, are shown in Table 5. It can be observed that the improvement moves are not able to reduce the number of trucks required. Nevertheless, there is a reduction in the number of pallets in the last truck, much more significant in Set I (257 pallets) than in Set II (10 pallets). As Table 4 showed, the numbers of trucks obtained by the randomized constructive algorithm are close to the values of the lower bounds, leaving little room for improvement, but reducing the number of pallets in the last truck is also of interest in the daily operation of the company.

6. Conclusions

The combined problem of putting products onto pallets and loading the pallets into trucks has received less attention in the literature than the standard Container Loading Problem, but it is a real problem faced by many distribution companies in their daily operations. In this study, we have taken the characteristics of the problem of a large logistics company as a reference, but it is common to many other companies. All the constraints defining the problem, especially those related to weight limits and weight distribution inside the trucks, have been considered.

Our approach has been to tackle the problem as a whole, without decomposing it into two different phases for pallet building and truck loading. The pallets are built taking into account the characteristics of the positions in which they will be loaded in the truck: horizontal dimensions, height, and weight, looking for maximum utilization of the trucks. As the problem has many specific constraints, we have developed a constructive algorithm tailored to the problem which provides quite good solutions in very short computing times. Nevertheless, as a simple constructive procedure is unable to obtain high-quality solutions for all the instances, we have developed a complete GRASP algorithm, including a randomization strategy for the constructive process and some improvement moves.

The results obtained by the GRASP algorithm have been compared with those obtained by some simple lower bounds on two sets of real instances involving up to 44 trucks. For more than 90% of the instances, the number of trucks obtained by the GRASP algorithm was the same as or included just one more truck than the lower bound. This indicates 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.

In our future work, we plan to extend this study to other distribution companies, to assess the suitability of the proposed algorithm for other related problems, possibly with some other specific constraints. Another interesting line of research would be to extend our algorithm to problems involving pallets and trucks of different types.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

This study has been partially supported by the Spanish Ministry of Science and Technology, DPI2011-24977 and DPI2014-53665-P, by Project PPII-2014-011-A, Consejería de Educación y Ciencia, Junta de Comunidades de Castilla-La Mancha, and by Generalitat Valenciana, PROM-ETEO/2013/049.

References

- [1] R. Morabito, S. R. Morales, and J. A. Widmer, "Loading optimization of palletized products on trucks," *Transportation Research Part E: Logistics and Transportation Review*, vol. 36, no. 4, pp. 285–296, 2000.
- [2] R. Morabito and S. Morales, "A simple and effective recursive procedure for the manufacturer's pallet loading problem," *Journal of the Operational Research Society*, vol. 49, no. 8, pp. 819–828, 1998.
- [3] S. Takahara, "Loading problem in multiple containers and pallets using strategic search method," in *Modeling Decisions for Artificial Intelligence*, V. Torra, Y. Narukawa, and S. Miyamoto, Eds., vol. 3558 of *Lecture Notes in Computer Science*, pp. 448–456, Springer, 2005.
- [4] R. W. Haessler and F. B. Talbot, "Load planning for shipments of low density products," *European Journal of Operational Research*, vol. 44, no. 2, pp. 289–299, 1990.
- [5] A. Moura and A. Bortfeldt, "A two-stage packing problem procedure," *International Transactions in Operational Research*, 2016.
- [6] A. Moura and J. F. Oliveira, "A GRASP approach to the container-loading problem," *IEEE Intelligent Systems*, vol. 20, no. 4, pp. 50–57, 2005.
- [7] E. E. Zachariadis, C. D. Tarantilis, and C. T. Kiranoudis, "The pallet-packing vehicle routing problem," *Transportation Science*, vol. 46, no. 3, pp. 341–358, 2012.
- [8] M. Alonso, R. Alvarez-Valdes, M. Iori, F. Parreño, and J. Tamarit, "Mathematical models for multi container loading problems," *Omega*, 2016.
- [9] A. Bortfeldt and G. Wäscher, "Constraints in container loading—a state-of-the-art review," *European Journal of Operational Research*, vol. 229, no. 1, pp. 1–20, 2013.
- [10] H. Gehring and A. Bortfeldt, "A genetic algorithm for solving the container loading problem," *International Transactions in Operational Research*, vol. 4, no. 5-6, pp. 401–418, 1997.
- [11] A. Bortfeldt, H. Gehring, and D. Mack, "A parallel tabu search algorithm for solving the container loading problem," *Parallel Computing*, vol. 29, no. 5, pp. 641–662, 2003.
- [12] J. Terno, G. Scheithauer, U. Sommerweiß, and J. Riehme, "An efficient approach for the multi-pallet loading problem," *European Journal of Operational Research*, vol. 123, no. 2, pp. 372–381, 2000.
- [13] J. Egeblad, C. Garavelli, S. Lisi, and D. Pisinger, "Heuristics for container loading of furniture," *European Journal of Operational Research*, vol. 200, no. 3, pp. 881–892, 2010.
- [14] E. E. Bischoff and M. D. Marriott, "A comparative evaluation of heuristics for container loading," *European Journal of Operational Research*, vol. 44, no. 2, pp. 267–276, 1990.
- [15] A. Bortfeldt and H. Gehring, "A hybrid genetic algorithm for the container loading problem," *European Journal of Operational Research*, vol. 131, no. 1, pp. 143–161, 2001.
- [16] A. Lim, H. Ma, C. Qiu, and W. Zhu, "The single container loading problem with axle weight constraints," *International Journal of Production Economics*, vol. 144, no. 1, pp. 358–369, 2013.
- [17] E. E. Bischoff and M. S. W. Ratcliff, "Issues in the development of approaches to container loading," *Omega*, vol. 23, no. 4, pp. 377–390, 1995.
- [18] J. Terno, G. Scheithauer, U. Sommerweiss, and J. Riehme, "An efficient approach for the multi-pallet loading problem," *European Journal of Operational Research*, vol. 123, no. 2, pp. 372–381, 2000.
- [19] L. Junqueira, R. Morabito, and D. Sato Yamashita, "Three-dimensional container loading models with cargo stability and

- load bearing constraints,” *Computers and Operations Research*, vol. 39, no. 1, pp. 74–85, 2012.
- [20] M. T. Alonso, R. Alvarez-Valdes, J. M. Tamarit, and F. Parreño, “A reactive GRASP algorithm for the container loading problem with load-bearing constraints,” *European Journal of Industrial Engineering*, vol. 8, no. 5, pp. 669–694, 2014.
- [21] A. Ramos, J. F. Oliveira, and M. P. Lopes, “A physical packing sequence algorithm for the container loading problem with static mechanical equilibrium conditions,” *International Transactions in Operational Research*, vol. 23, no. 1-2, pp. 215–238, 2016.
- [22] O. C. B. De Araújo and V. A. Armentano, “A multi-start random constructive heuristic for the container loading problem,” *Pesquisa Operacional*, vol. 27, no. 2, pp. 311–331, 2007.
- [23] T. Fanslau and A. Bortfeldt, “A tree search algorithm for solving the container loading problem,” *INFORMS Journal on Computing*, vol. 22, no. 2, pp. 222–235, 2010.
- [24] A. G. Ramos, J. F. Oliveira, J. F. Gonçalves, and M. P. Lopes, “Dynamic stability metrics for the container loading problem,” *Transportation Research Part C: Emerging Technologies*, vol. 60, pp. 480–497, 2015.
- [25] M. Iori, J.-J. Salazar-González, and D. Vigo, “An exact approach for the vehicle routing problem with two-dimensional loading constraints,” *Transportation Science*, vol. 41, no. 2, pp. 253–264, 2007.
- [26] A. Bortfeldt, “A hybrid algorithm for the capacitated vehicle routing problem with three-dimensional loading constraints,” *Computers and Operations Research*, vol. 39, no. 9, pp. 2248–2257, 2012.
- [27] L. Junqueira, J. F. Oliveira, M. A. Carravilla, and R. Morabito, “An optimization model for the vehicle routing problem with practical three-dimensional loading constraints,” *International Transactions in Operational Research*, vol. 20, no. 5, pp. 645–666, 2013.
- [28] S. Ceschia, A. Schaerf, and T. Stützle, “Local search techniques for a routing-packing problem,” *Computers and Industrial Engineering*, vol. 66, no. 4, pp. 1138–1149, 2013.
- [29] Y. Tao and F. Wang, “An effective tabu search approach with improved loading algorithms for the 3L-CVRP,” *Computers & Operations Research*, vol. 55, pp. 127–140, 2015.
- [30] L. Junqueira and R. Morabito, “Heuristic algorithms for a three-dimensional loading capacitated vehicle routing problem in a carrier,” *Computers and Industrial Engineering*, vol. 88, pp. 110–130, 2015.
- [31] K. F. Doerner, G. Fuellerer, R. F. Hartl, M. Gronalt, and M. Iori, “Metaheuristics for the vehicle routing problem with loading constraints,” *Networks*, vol. 49, no. 4, pp. 294–307, 2007.
- [32] H. Pollaris, K. Braekers, A. Caris, G. Janssens, and S. Limbourg, “Capacitated vehicle routing problem with sequence-based pallet loading and axle weight constraints,” *EURO Journal on Transportation and Logistics*, 2014.
- [33] M. Iori and S. Martello, “Routing problems with loading constraints,” *TOP*, vol. 18, no. 1, pp. 4–27, 2010.
- [34] M. Iori and S. Martello, “An annotated bibliography of combined routing and loading problems,” *Yugoslav Journal of Operations Research*, vol. 23, no. 3, pp. 311–326, 2013.
- [35] H. Pollaris, K. Braekers, A. Caris, G. K. Janssens, and S. Limbourg, “Vehicle routing problems with loading constraints: state-of-the-art and future directions,” *OR Spectrum*, vol. 37, no. 2, pp. 297–330, 2015.
- [36] SCANIA, April 2016, <https://bodybuilderhomepage.scania.com/>.
- [37] M. N. Arenales, A. C. Cherri, D. N. Nascimento, and A. Vianna, “A new mathematical model for the cutting stock/leftover problem,” *Pesquisa Operacional*, vol. 35, no. 3, pp. 509–522, 2015.
- [38] R. Andrade, E. G. Birgin, and R. Morabito, “Two-stage two-dimensional guillotine cutting stock problems with usable leftover,” *International Transactions in Operational Research*, vol. 23, no. 1-2, pp. 121–145, 2016.
- [39] G. Wäscher, H. Haußner, and H. Schumann, “An improved typology of cutting and packing problems,” *European Journal of Operational Research*, vol. 183, no. 3, pp. 1109–1130, 2007.
- [40] M. G. C. Resende and R. F. Werneck, “A hybrid heuristic for the p-median problem,” *Journal of Heuristics*, vol. 10, no. 1, pp. 59–88, 2004.
- [41] M. Prais and C. C. Ribeiro, “Reactive grasp: an application to a matrix decomposition problem in tdma traffic assignment,” *INFORMS Journal on Computing*, vol. 12, no. 3, pp. 164–176, 2000.
- [42] R. Marti and J. Moreno-Vega, “Metodos multiarranque,” *Inteligencia Artificial*, vol. 19, no. 2, pp. 49–60, 2003.
- [43] ORTEC, February 2016, <http://ortec.com/>.