INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH

WILEY

INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH

Intl. Trans. in Op. Res. 24 (2017) 43–58
DOI: 10.1111/itor.12251

A two-stage packing problem procedure

Ana Moura[a,b] and Andreas Bortfeldt[c]

[a] *Universidade de Aveiro, Aveiro, Portugal*
[b] *CIDMA—Center for Research and Development in Mathematics and Applications, Aveiro, Portugal*
[c] *Otto-von-Guericke-University Magdeburg, Magdeburg, Germany*
E-mail: ana.moura@ua.pt [Moura]; andreas.bortfeldt@ovgu.de [Bortfeldt]

Received 26 May 2015; received in revised form 25 September 2015; accepted 24 November 2015

Abstract

This paper deals with a two-stage packing problem that has to be solved in the daily distribution process of a Portuguese trading company. In the first stage, boxes including goods are to be packed on pallets, while in the second stage these pallets are loaded into one or more trucks. The boxes have to be transported to different customers, and the main goal is to guarantee a sufficient utilization of the truck loading space. A two-stage packing procedure is proposed to cover both problem stages. First, boxes are loaded onto pallets by means of a well-known container loading algorithm. Then, trucks are filled with loaded pallets by a tree search algorithm. The performance of the two-stage approach was evaluated using a set of instances that are based on actual company data.

Keywords: bin packing; pallet loading; GRASP; tree search

1. Introduction

Managing the distribution of goods is a vital operation that has a major economic impact. Clients' satisfaction depends mainly on meeting their demand as effectively as possible. This is commonly described as providing a service to a client. Usually, a service is a mixture of different distribution characteristics, for example, product availability, delivery time, delivery programming, and good condition after delivery (cf. Moura and Oliveira, 2009).

This paper deals with a two-stage packing problem that occurs in the daily distribution process of the Portuguese company Oliveira & Irmão (O&I). O&I delivers sanitary articles to foreign customers by trucks. Each day, some small groups of (say) three or four customers, located in the same European region (e.g. the Netherlands), are visited on the same route.

The goods ordered by a customer group are packaged in boxes that must be loaded in one or more trucks. Trucks are not loaded with boxes directly. Instead, first identical pallets are packed with boxes and then these pallets are loaded into trucks (two-stage packing). The pallets are packed with boxes by means of robots at the end of the production line. Thus, packing patterns must contain only goods for one customer and only boxes of the same type, that is, with same side dimensions. In the second packing stage, stacking of pallets within trucks is allowed. Each truck offers the same number of stacking places with fixed positions. Several side constraints have to be observed, for example, the so-called LIFO (last-in, first-out) constraint. It stipulates that no reloading of boxes must take place at the different customer sites of a given truck route. The packing operations of both stages have to be done in such a way that the truck loading spaces are utilized best, that is, the unused loading space has to be minimized.

Generating high-quality solutions for the drafted two-stage packing problem in short time is crucial for an efficient distribution process at O&I. If an applied solution procedure is, nevertheless, not able to achieve a good utilization of trucks since "the data do not match," a dispatcher could try to negotiate with customers. She or he could propose, for example, to postpone some parts of the daily orders to another day. In that way a set of orders could be achieved for the given day that allows for well-utilized trucks.

Little attention has been paid so far to this packing problem in the literature; thus, this paper tries to develop an effective solution procedure for solving it.

The remainder of the paper is structured as follows: in Section 2 the two-stage packing problem mentioned above is formulated in detail. In Sections 3 and 4, solution procedures for the two sub-problems are specified. Section 5 reports computational results. Finally, in Section 6 some conclusions are drawn.

2. The two-stage packing problem

The two-stage packing problem occurring in the distribution of goods for foreign customers who are served by trucks is now defined with all details. Let us assume that the following entities are given: (a) A single truck type with a rectangular loading space and a single pallet type; the numbers of available trucks and pallets are not limited. (b) A route (i.e. sequence) of customers to be served and one order per customer. Each order comprises one or more box types and a corresponding number of demanded boxes per type. It is required to pack all boxes on pallets and then load the pallets in a minimal number of trucks. Moreover, all side constraints that are specified for both packing stages have to be observed. When boxes are packed onto pallets, the following constraints should be met:

(C1) Each pallet can be packed only with boxes of a single customer. Furthermore, each pallet can be packed only with boxes of the same type, that is, same side dimensions.

(C2) Each box must lie completely on its pallet, that is, it is not allowed that a box does protrude over the lateral borders of its pallet.

(C3) The loading height of a pallet is given by the distance between the highest top area of a box and the bottom of the pallet (the height of an empty pallet is viewed as part of the loading height). The loading height must not exceed a given maximum which is stipulated individually per customer.

(C4) The bottom area of each box has to be fully supported (i.e. to 100%) from below by the bottom area of the pallet or by the top areas of other boxes.

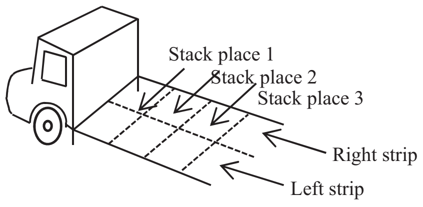

Fig. 1. Organization of a truck loading space.

The rectangular loading space of a truck is organized in two strips, called “left strip” and “right strip” (see Fig. 1), with equal dimensions. Each strip consists of *nStPl* stack places, where stack place 1 lies at the driver’s cabin, while stack place *nStPl* of a strip lies at the rear (near to the door) of the loading space. At each stack place, several pallets can be stacked on top of each other. A pallet is always to set on a stack place in such a way that the pallet length lies parallel to the width of the appropriate strip. Thus, the position of a pallet can be described unambiguously by four indices: index of truck (1,2, . . .), index of strip (left or right), index of stack place (1, . . . , *nStPl*), and index of position in stack (1,2, . . .), where value 1 means that the pallet lies on the bottom of a stack.

When pallets are loaded into one or more trucks, the following constraints should be satisfied:

(C5) The height of a stack of one or more pallets is defined as the sum of the loading heights of all pallets and it must not exceed the height of the truck loading space (truck height).

(C6) The total top area of a loaded pallet is defined as the sum of the top areas of all boxes whose top area equals the loading height of the pallet (see C3). The support rate *supTopArea* of a loaded pallet *A* is given as the quotient (*total top area of pallet A*)/(*bottom area of pallet type*) (in %). A further pallet may only be stacked upon pallet A if *supTopArea*(*A*) is not smaller than a given support rate *minSupTopArea* (in %). Otherwise pallet A is called a cap pallet and has to be packed on the top of a stack.

(C7) The boxes and pallets have to be stowed in such a way that no reloading of boxes is necessary at the different customer locations of a given truck route (LIFO constraint). This constraint shows two aspects. Let customers *A* and *B* be served in the same route and let customer *A* be visited first. Then a pallet with boxes for customer *B* must not lie upon a pallet for customer *A* in the same stack of a truck. Moreover, if a pallet for customer *A* is packed, then a pallet for customer *B* must not be packed on a stacking place with higher index in any strip of the same truck (cf. Fig. 2).

Since all given boxes of all orders have to be packed and the number of trucks is to be minimized, the two-stage packing problem can be viewed as a variant of the three-dimensional bin packing problem (3D-BPP). Additionally, a second optimization criterion is introduced for the problem of loading pallets into trucks (LPIT): if two solutions need the same number of trucks, the solution with the lower maximum stack height is considered the better one. Obviously, this criterion will strengthen the load stability of generated packing plans.

```
lbop(in: set of all boxes of all daily orders, out: plans of all packed pallets)
    for each customer do
        for each box type do
            while not yet packed boxes of current customer/box type
                  combination still available do
                  load another pallet with remaining boxes of current
                    customer/box type by means of CHMO
            endwhile
        endfor
    endfor
end.
```

Fig. 2. The LBOP procedure.

3. Literature review

Till now, only few papers have been published dealing with the problem studied here, that is, of packing goods on pallets and pallets on trucks. Morabito et al. (2000) address a 2D variant of the problem since the boxes cannot be stacked. In the pallet loading phase, a maximum number of goods are to be packed on each used pallet. In the following truck loading phase, the pallets are loaded in the minimum number of trucks. For both phases, the five-block algorithm proposed by Morabito and Morales (1998) is used. Takahara (2005) deals with a packing problem in which multiple containers and multiple pallets of different sizes are to be packed simultaneously by given rectangular goods. He proposes a heuristic in which solutions are represented by a permutation of the goods and a second permutation of the containers and pallets; near-optimal permutations are then determined by local search techniques.

Zachariadis et al. (2012) tackle the so-called pallet packing vehicle routing problem with time windows (PPVRPTW), that is, an integrated vehicle routing and 3D loading problem. In the PPVRPTW, goods of customers are first loaded onto identical pallets before the pallets are loaded into vehicles. The goal is to construct least cost routes for a set of customers with given 3D demands. So the packing part is similar to our problem, but it is combined with a routing problem. Nevertheless, there are differences in the packing parts of the problems, too. For example, the PPVRPTW does not include any type of LIFO constraint and pallets cannot be stacked on top of each other.

Alonso et al. (2015) deal with a two-stage packing problem in which pallets are built in the first stage and trucks are filled with pallets in the second stage. As in the problem at hand, the goal is to minimize the number of used trucks. Several side constraints are considered, for example, a weight limit for the cargo as well as a limit for the weight an axle can bear. The main difference to the problem considered in this paper is that pallets are packed with horizontal layers that are lying upon each other. These layers are constructed “in advance.” Thus, the pallet building problem becomes a 1D packing problem. The authors develop several integer linear programming (ILP) models to solve the resulting multi-container loading problem and test their approach by a large set of benchmark instances provided by a distribution company.

In the problem at hand, we have to first deal with loading boxes onto pallets (LBOP). This task can be reduced to a series of single container loading problems (SCLP). The SCLP requires packing a subset of a given set of 3D rectangular pieces into a single rectangular container so that the volume utilization is as large as possible. As the SCLP is NP-hard in the strict sense, most of the solution methods that were proposed in recent years are heuristics and metaheuristics. Moura and Oliveira (2005) suggested a GRASP algorithm for the SCLP, based on a wall-building constructive heuristic and achieve high levels of cargo stability without compromising the container volume utilization. Parreño et al. (2007, 2010a) also specified a GRASP approach and a variable neighborhood search algorithm. Fanslau and Bortfeldt (2010) developed a tree search heuristic. Gonçalves and Resende (2012) published a genetic algorithm. Zhu and Lim (2012) suggested a Greedy–Lookahead algorithm and Araya and Riff (2014) proposed a beam search approach.

Since the whole problem can be understood as a variant of the 3D-BPP and the second problem stage can be reduced to a 1D-BPP (cf. Section 5), we will also report the 1D and 3D bin packing solution approaches. The BPP with one type of container requires packing a set of items in the least number of bins and is an NP-hard problem independent of its dimensionality. In the 3D case, mostly metaheuristics were developed. Faroe et al. (2003) propose a guided local search heuristic, while Crainic et al. (2008, 2009) suggested extreme point-based heuristics and a two-level tabu search method. These methods belong to the strongest solution approaches for the 3D-BPP. Mack and Bortfeldt (2012) presented a heuristic for solving large 3D-BPP instances, and Parreño et al. (2010b) developed a hybrid approach for the 3D-BPP. In the 1D case, heuristics and metaheuristics as well as exact approaches belong to the successful solution methods. Exact approaches are provided in Martello and Toth (1990), Scholl et al. (1997), Schwerin and Wäscher (1999), and Schoenfield (2002). Heuristic solution methods were suggested by Falkenauer (1996), Fleszar and Hindi (2002), and Alvim et al. (2004), to mention a few.

The reader is referred to Wäscher et al. (2007) for an up-to-date typology of cutting and packing problems. A survey of 3D container loading solution methods, focusing on the inclusion of relevant constraints, is given by Bortfeldt and Wäscher (2013). A very recent overview of solution methods for the pallet loading problem is presented by Silva et al. (2014).

The working paper by Moura and Bortfeldt (2010) is a preliminary version of this paper; it has been significantly extended regarding numerical experiments and other topics in this paper.

4. Loading boxes onto pallets

Our solution approach to the two-stage packing problem includes two separate procedures. By means of the first procedure, called LBOP, all boxes are packed on pallets, and by means of the second procedure, called LPIT, all pallets are loaded in a minimum number of trucks. The procedures are called consecutively and there is no feedback, that is, after the pallets were packed they are no longer repacked.

Our approach for the first problem stage is to solve a series of container loading problems in order to pack all given boxes of all customers on pallets.

The GRASP method by Moura and Oliveira (2005) achieves good volume utilization and proved to be very sensitive in terms of load stability. Last but not least, solutions are provided very rapidly, and this could encourage the O&I dispatchers to apply the planned decision support tool directly when negotiating with customers to perform what-if-computations, etc. For these reasons, the LBOP procedure was derived from the GRASP method by Moura and Oliveira (2005) in the following manner.

The GRASP method includes a constructive heuristic, denoted here by CHMO, and a local search algorithm. However, only CHMO is used for the LBOP procedure, while the local search algorithm is ignored. A standard parameterization of heuristic CHMO, applied by Moura and Oliveira (2005), has been adopted. Now, the LBOP procedure is built around CHMO, as shown in Fig. 2. By means of both "for" loops, it is ensured that boxes belonging to different customers and/or box types are loaded onto different pallets, that is, constraint (C1) is met.

To pack the boxes of one customer and one box type, the heuristic CHMO is generally applied several times, that is, several pallets are packed until all currently relevant boxes are exhausted. Since CHMO is a container loading procedure, the constraints (C2) and (C3) are automatically satisfied if the maximum loading height of a pallet is taken as the container height. Moreover, CHMO is able to observe the support constraint (C4) without modification.

In the first packing stage, there are two critical success factors. First, the mean volume utilization of all packed pallets should be as large as possible. The volume utilization of a single loaded pallet is computed as quotient (*total volume of loaded boxes*)/(*volume of pallet space*) (in %), where the pallet space volume is given as product (*pallet bottom area*) × (*loading height – height of empty pallet*). Furthermore, the number of generated cap pallets (see Section 2, C6) should be as small as possible, since a higher number of cap pallets will lead to a higher number of used stacks. Later, we will see whether and how the LBOP procedure considers these critical success factors.

5. Loading pallets into trucks

All pallets that were packed with boxes have to be loaded then into the minimum number of trucks. Furthermore, the constraints (C5) to (C7) and the second optimization criterion—minimization of the maximum stack height—must also be taken into account. In Table 1, the input and output data of the second packing stage are listed. Each packed pallet (that is input to the second stage) is given by its loading height, its total top area, and the corresponding customer. Each stacked pallet

Table 1
Input and output data of second packing stage

Input data
nC: Number of customers
Sequence of customers
hT: truck height
Pallet bottom area
nStPl: Number of stack places per strip
minSupTopArea: required support rate per pallet
nP: Number of packed pallets
Set of packed pallets
Output data
nTrucks: Number of loaded trucks
maxStackHeight: maximum stack height
Set of stacked pallets

(output of the second stage) is defined by a pallet index and by four indices regarding truck, strip, stack place, and position in stack (see Section 2).

The problem of LPIT considered here can be viewed as a special variant of 1D-BPP in which the stack places act as bins (with truck height as bin size) and the heights of the packed pallets act as small pieces. Therefore, a tree search procedure was devised for the 1D-BPP.

5.1. *Schema of tree search*

Solutions are constructed stepwise, and in each step a solution is extended by stacking one further pallet. A depth-first search is applied, that is, the next solution to be extended is one of the existing partial solutions with the largest number of pallets already stacked. The search is carried out by means of a recursive function, stackOnePallet() (shown in Fig. 3), that takes a partial solution and the corresponding number of stacked pallets as arguments. If the solution is already complete, the best solution so far is only updated where necessary. Otherwise, the current partial solution is extended by stacking another pallet in alternative variants (see below), and the function stackOnePallet() is called again for each of the variants.

To reduce the redundancy of the search, the order in which the pallets are stacked is fixed. Before the function stackOnePallet() is called for the first time, the packed pallets are sorted according to the visiting sequence of the customers. Pallets that belong to customers to be visited last are stacked first (near the driver's cabin) and vice versa. Pallets of the same customer are sorted according to the following criteria. A cap pallet has to be stacked before a noncap pallet. This seems to be implausible at first glance. However, stacking cap pallets of a customer first will lead to additional places for these pallets on the top of previously loaded pallets of other customers.

It was found in preliminary experiments that results in terms of truck number for the chosen loading order of pallets are significantly better than for the order where cap pallets are stacked after noncap pallets of the same customer. If both pallets are either cap pallets or noncap pallets, then the pallet with greater loading height has to be stacked first. For the next pallet to be stacked, all feasible variants are determined to place this pallet on top of an existing stack (with one or more pallets). Additionally, a new stack is opened in front of the existing ones and the current pallet is placed there. This procedure corresponds to the branching schema applied to the 1D-BPP by Martello and Toth (1990). However, the number of tested stacking variants *nStVar* is limited by a

```
stackOnePallet(inout: nStackedPallets, solution)
    if nStackedPallets ≥ nP then  // solution complete
        update best solution where necessary; return
    Endif
    generate nStVar stacking variants for next pallet
    for i := 1 to nStVar do
        extend solution by i-th stacking variant and get newSolution
        stackOnePallet(nStackedPallets+1, newSolution)
    endfor
end.
```

Fig. 3. Recursive function stackOnePallet().

parameter *maxSucc*. All feasible stacking variants are sorted according to the residual height left in the filled stack in ascending order. Then only the first *maxSucc* stacking variants with the lowest residual heights (in the stacks just filled) are considered. In this way, stacking variants with highly utilized stacks are favored.

To limit the total effort of the procedure, a time limit is used. Furthermore, some bounds of the objective function serve to shorten the search.

5.2. Bounding the search

Two lower bounds are derived for the number of needed trucks. The first one (called static bound) refers to a problem instance without any additional information. The second one (called dynamic bound) is based on a partial solution with one or more stacked pallets. Similarly, a static and a dynamic lower bound is derived for the second objective criterion, namely, the maximum stack height. The static lower bounds are used each time a new complete solution was generated. In this case, the search is aborted if the complete solution equals both of the lower bounds, that is, if a globally optimal solution was reached. The dynamic lower bounds are used each time before the current solution is extended. An extension is saved if the best solution so far cannot be improved any longer.

5.2.1. Static bounds

The following notation is agreed (see also Table 1). Let the customers of one route be indexed from 1 to nC, so that a customer, c, is visited as the cth customer. Let $hp(c)$ denote the total loading height of all pallets of customer c and let $ncp(c)$ be the number of cap pallets of customer c ($c = 1, \ldots, nC$). Finally, let $nst1(c) := \Sigma_{i>c} ncp(i)$ and $nst2(c) := \lceil \Sigma_{i \le c} hp(i)/hT \rceil$, $c = 0, 1, \ldots, nC$.

Proposition 1.

(i) *$LBnSt(c) := nst1(c) + nst2(c)$ is a lower bound on the number of filled stacks, $c = 0, 1, \ldots, nC$.*
(ii) *$LBnSt := max\{LBnSt(c),\ c = 0, 1, \ldots, nC\}$ is a lower bound on the number of filled stacks.*
(iii) *$LBnTrucks := \lceil \Sigma LBnSt/(2 \times nStPl) \rceil$ is a lower bound on the number of needed trucks.*

Proof. (i) Since two cap pallets cannot be packed into one stack, at least $nst1(c)$ stacks are needed for the customers $i = c+1, \ldots, nC$. Packing a pallet of a customer $j \le c$ and a cap pallet of a customer $i > c$ into one stack would violate one of the constraints (C7) or (C8). Hence, at least $nst2(c)$ additional stacks are necessary to load all pallets. The special cases $c = 0$ and $c = nC$ are trivial. (ii) and (iii) are obvious. □

A lower bound of the maximum stack height is formulated for a given number of trucks in a complete solution. Let *maxhp* denote the maximum loading height of a packed pallet.

Proposition 2. *$LBmaxStHeight := max\{maxhp,\ \lceil \sum_{c=1}^{nC} hp(c)/(2 \times nTrucks \times nStPl) \rceil\}$ is a lower bound of the maximum stack height for all solutions with nTrucks trucks.*

Proof. Obvious. □

5.2.2. Dynamic bounds

Now let a partial solution with *nStacks* used stacks and *nPallets* (*nPallets*<*nP*) loaded pallets be given. Hence, the pallets with indices *nPallets*+1, . . . ,*nP* are free, that is, still to be packed. The total pallet height of all free pallets is denoted as *hpres*. An existing (partially filled) stack is called usable if it can still accommodate at least one free pallet. Let *hstres* be the sum of the residual heights of all usable stacks.

Proposition 3.

(i) *LBnAddStacks* $:= max\{\lceil (hpres\text{-}hstres)/hT \rceil, 0\}$ *is a lower bound of the number of additional stacks needed to load the residual pallets.*
(ii) *LBnTrucksD* $:= \lceil (nStacks+LBnAddStacks)/(2 \times nStPl) \rceil$ *is a lower bound of the number of trucks needed to load all pallets.*

Proof. (i) Note that the height difference *hpres – hstres* represents a lower bound of the total pallet height that cannot be accommodated in existing stacks. (ii) Obvious. □

A partial solution with *nStacks* used stacks and *nPallets* (*nPallets*<*nP*) loaded pallets is assumed. A lower bound of the maximum stack height is formulated for all complete solutions with *nTrucks* trucks. Let *maxhst* denote the maximum height of all used stacks. *nStacksRes* stands for the difference *nTrucks***nStPl* – *nStacks*, that is, the maximum number of additional stacks.

Proposition 4. *LBmaxStHeightD* $:= max\{maxhst, maxhp, \lceil (hpres - hstres)/nStacksRes \rceil\}$ *is a lower bound of the maximum stack height for all solutions with nTrucks trucks.*

Proof. Obvious, cf. proof of Proposition 3. □

5.3. An ILP model to load pallets into trucks

In order to evaluate the LPIT approach, an ILP model to load pallets into trucks was developed. The constants of the model are shown below:

- m = number of pallets to be loaded;
- h_i = height of pallet i ($i = 1, \ldots, m$);
- cap_i = cap pallet indicator; $cap_i = 1$, if pallet i is a cap pallet; 0, otherwise ($i = 1, \ldots, m$);
- p_i = priority index of pallet i, $p_i = p$, if the customer of pallet i is visited as pth customer in its tour ($i = 1, \ldots, m$);
- p_{max} = maximum priority index (“Big-M”);
- H = height of a truck;
- nst = number of stack places within the left or right strip of a truck;
- nt = number of available trucks; nt is set to $\lceil m/(2nst) \rceil$, being an upper bound of needed trucks.

The decision variables are as follows:

- $x_{tij} = 1$, if pallet i is stored at stack j in truck t, and 0 if not ($t = 1, \ldots, nt$; $i = 1, \ldots, m$ and $j = 1, \ldots, 2nst$);

- $y_{tj} = 1$, if stack j of truck t is filled by at least one pallet, and 0 if not ($t = 1, \ldots, nt$ and $j = 1, \ldots, 2nst$).

The objective function is defined as

$$\min \sum_{t=1}^{nt} \sum_{j=1}^{2nst} y_{tj} \tag{1}$$

The loading model minimizes the number of used stacks in all trucks. This objective function (1) is subject to several constraints:

$$\sum_{t=1}^{nt} \sum_{j=1}^{2nst} x_{tij} = 1, \qquad \forall i = 1, \ldots, m \tag{2}$$

$$y_{tj} \geq y_{tj+1}, \qquad \forall t = 1, \ldots, nt; \forall j = 1, \ldots, 2nst - 1 \tag{3}$$

$$y_{tj} \geq y_{(t+1)1}, \qquad \forall t = 1, \ldots, nt - 1 \tag{4}$$

$$\sum_{i=1}^{m} h_i \times x_{tij} \leq y_{tj} \times H, \quad \forall t = 1, \ldots, nt; \forall j = 1, \ldots, 2nst \tag{5}$$

$$\sum_{i=1}^{m} cap_i \times x_{tij} \leq y_{tj}, \qquad \forall t = 1, \ldots, nt; \forall j = 1, \ldots, 2nst \tag{6}$$

$$\begin{aligned} &cap_i \times p_i \times x_{tij} \leq p_k \times x_{tkj} + p_{\max}(1 - x_{tkj}), \\ &\qquad \forall t = 1, \ldots, nt; \forall i, k = 1, \ldots, m; \forall j = 1, \ldots, 2nst \end{aligned} \tag{7}$$

$$\begin{aligned} &p_i \times x_{ti(2l-1)} + p_{\max}(1 - x_{ti(2l-1)}) \geq p_k \times x_{tk(2l+1)}, \\ &p_i \times x_{ti(2l-1)} + p_{\max}(1 - x_{ti(2l-1)}) \geq p_k \times x_{tk(2l+2)}, \\ &p_i \times x_{ti(2l)} + p_{\max}(1 - x_{ti(2l)}) \geq p_k \times x_{tk(2l+1)}, \\ &p_i \times x_{ti(2l)} + p_{\max}(1 - x_{ti(2l)}) \geq p_k \times x_{tk(2l+2)}, \\ &\qquad \forall t = 1, \ldots, nt; \forall i, k = 1, \ldots, m \wedge i \neq k; \forall l = 1, \ldots, nst - 1 \end{aligned} \tag{8}$$

Constraint (2) ensures that each pallet is assigned to exactly one stack. Constraints (3) and (4) guarantee that a higher indexed stack can be used only if the previous one was used and a new truck can be used only if the last stack of the previous one is already used. The next constraint (5) is to ensure that the maximum stack height (truck height) is not exceeded. Constraint (6) warrants that only one cap pallet can be assigned to a stack and only used stacks may be considered. Inequalities (7) and (8) are related to the LIFO constraint. Inequality (7) relates to the LIFO constraint within each stack and ensures that the priority of a cap pallet is smaller than the priority of any other pallet in the same stack. If a stack has no cap pallet, then all pallets can be placed with falling priority from bottom to top. The set of inequalities (8) serves to ensure the LIFO constraint between pairs of stacks belonging to the same truck. These inequalities guarantee that a pallet A that is situated nearer to the door than a pallet B has a lower (or same) priority compared with pallet B (cf. Fig. 1).

6. Computational test

The solution procedure for the considered two-stage packing problem was implemented in C and tested by means of an Intel PC (Core i5-2320 3.3 GHz). In this section, four sets of test instances are defined before corresponding computational results are reported and commented.

6.1. Four sets of test instances

The first set includes 12 instances of the (complete) two-stage packing problem and these instances were taken from the distribution process of company O&I. In Table 2 the main characteristics of the 12 instances are listed.

The maximum loading height of a pallet is stipulated individually per customer. Figure 4 shows some pallets that were loaded using different limits of the loading height.

The second set also includes 12 instances. The data of these instances are as in the first set, except that the number of boxes per type is multiplied by three. This means more trucks are needed to transport all boxes. The third set of test instances comprises 126 instances of the packing problem of the second packing stage. Table 3 shows the main characteristics of these relatively large instances that serve to subject the LPIT procedure to a more thorough test.

Table 2
Characteristics of the 12 instances of set 1 (all dimensions in centimeters)

Characteristic	Data
Truck loading space	l: 800, w: 250, h: 260
Pallet type (empty pallet)	l: 120, w: 80, h: 15
Stack places per strip *nStPl*	10
Requested pallet support *minSupTopArea*	70%
Number of routes per instance	1 (constantly)
Number of customers per instance	3–5
Number of demanded box types per customer	1–4
Mean number of boxes per instance	5648.1

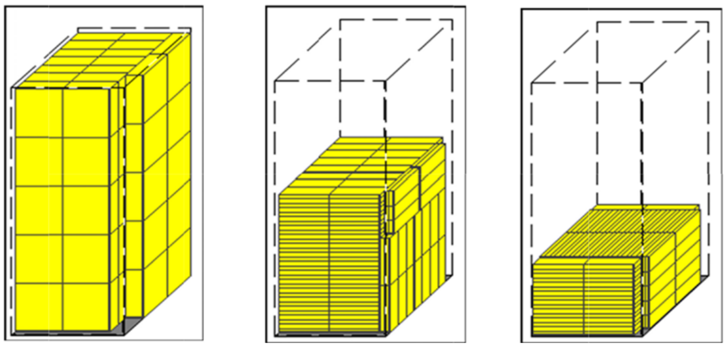
Fig. 4. Examples of loaded pallets.

Table 3
Characteristics of the 126 instances of set 3 (all dimensions in centimeters)

Characteristic	Data
Truck height (height of loading space)	230
Stack places per strip *nStPl*	10
Requested pallet support *minSupTopArea*	80%
Number of customers per instance	1–3
Number of pallets per instance	51–105
Pallet heights (as rate of truck height)	16–33%
Percentage of cap pallets (see Section 2)	9–19

Table 4
Results for the 12 instances of set 1, first packing stage

Instance	Customers, *n*	Loaded pallets, *n*	Mean volume utilization (%)	Cap pallets, *n*	Rate of cap pallets (%)
1	5	40	94	0	0.0
2	4	50	90	5	10.0
3	5	60	91	5	8.3
4	4	40	93	0	0.0
5	3	49	89	10	20.4
6	4	60	86	8	13.3
7	5	48	89	4	8.3
8	3	40	90	0	0.0
9	4	47	94	0	0.0
10	5	50	91	6	12.0
11	5	52	93	10	19.2
12	5	51	91	6	11.8
Average	–	**48.9**	**91**	**4.5**	**8.6**

The fourth set of instances includes 12 instances with the same characteristics as set 3 with two exceptions. Now the number of pallets is constantly 180 and the pallet heights vary between 33% and 50% of the truck height.

6.2. Computational results

The number of iterations for the LBOP procedure was set to 100. The time limit for the LPIT procedure was set to 3 seconds for sets 1 to 3, and to 30 seconds for set 4. The number of stacking variants was bounded by $maxSucc = 5$ in any case.

In Table 4, the results for the 12 instances of set 1 are shown that were achieved in the first packing stage by the LBOP procedure.

The mean running time for each instance is 43 seconds. The following data are shown per instance: number of customers, total number of loaded pallets, mean volume utilization over all pallets (in %), and number and rate (in %) of generated cap pallets. Considering the critical success factors, namely, the mean volume utilization and the number (or rate) of cap pallets, it seems that the LBOP

Table 5
Results for the 12 instances of set 1, second packing stage

Instance	Customers, *n*	Pallets, *n*	Stacks, *n*	Trucks, *n*	Truck span	Maximum stack height	Height gap (%)
1	5	40	20	1	0	256	0.4
2	4	50	37	2	1	170	31.8
3	5	60	20	1	0	251	2.0
4	4	40	20	1	0	256	0.4
5	3	49	39	2	1	164	27.1
6	4	60	20	1	0	249	1.6
7	5	48	39	2	1	165	27.9
8	3	40	20	1	0	256	0.4
9	4	47	37	2	1	166	28.7
10	5	50	38	2	1	168	30.2
11	5	52	39	2	0	170	31.8
12	5	51	40	2	1	165	27.9
Average	–	**48.9**	**30.7**	**1.5**	**0.5**	–	**17.6**

procedure yields sufficient results. In particular, the numbers of cap pallets do not lead to increasing stack numbers.

Table 5 contains the results for the 12 instances of set 1 that were achieved in the second packing stage by the LPIT procedure. For each instance, the following data are indicated: number of customers, number of pallets to be packed, number of stacks used, number of loaded trucks (*nTrucks*), truck span, maximum stack height (*maxStackHeight*, in cm), and height gap. The truck span is given as difference (*nTrucks – LbnTrucks*) (see Section 4). The height gap is computed as ratio (*maxStackHeight – LBmaxStHeight*)/*LBmaxStHeight* (in %, see Section 4); stack heights are given in centimeters.

The main result is that in 6 of 12 instances, the optimal truck number was reached and that it was exceeded by only one truck in the other instances. The height gaps are moderate: they do never exceed 32% and amount to 17.6% on average. It should be stressed that the results are compared only with lower bounds (that are more or less weak), while the optimal truck number is unknown for 6 of 12 instances and the optimal stack height is unknown in all cases.

A further result of the test of set 1 is that the computational effort per instance is rather low. The mean total computing time over the 12 instances is around 46 seconds. The main part of the computing time is needed to load pallets in the first stage, while the time for the second stage never exceeds three seconds.

In order to better evaluate the quality of the LPIT procedure, the ILP model (see Section 5.3) was implemented in CPLEX, and experiments were run on an Intel CORE i7 vPro with 2.2 GHz and 8 Gb of memory. The time limit for the computation was set to two hours per instance. Table 6 presents the results for the 12 instances of set 1 achieved with the ILP model.

The first five columns are as in Table 5; the last column shows the CPU time in seconds. Optimal solutions were generated for all 12 instances of set 1. A comparison of the truck numbers in Tables 5 and 6 proves that the LPIT procedure (also) reaches the optimal truck number for all instances. Thus, it seems that the LPIT procedure is a suitable approach to solve the second packing stage. On

Table 6
Results for the 12 instances of set 1, second packing stage, ILP model

Instance	Customers, *n*	Pallets, *n*	Stacks, *n*	Trucks, *n*	CPU time (seconds)
1	5	40	20	1	4.3
2	4	50	35	2	5132.8
3	5	60	20	1	32.4
4	4	40	20	1	5.1
5	3	49	39	2	6012.4
6	4	60	20	1	108.2
7	5	48	39	2	4286.2
8	3	40	20	1	4.0
9	4	47	37	2	6300.2
10	5	50	32	2	1297.4
11	5	52	36	2	4432.3
12	5	51	38	2	5486.2
Average	–	**48.9**	**29.6**	**1.5**	**2758.5**

the other hand, the proposed ILP model is very time consuming and is therefore not suitable for practical application.

Regarding sets 2–4 of test instances, only overall results are reported here. For the 12 instances of *set 2* the numbers of generated pallets for each combination customer/box type, of course, tripled compared with set 1. In the second packing stage, the optimal number of trucks was reached six times, four times the (static) lower bound of the truck number *LBnTrucks* was missed by one truck, and only two times it was exceeded by two trucks. The average height gap amounts to 19%.

For the 126 LPIT instances of *set 3,* good results were obtained. Based on the (static) lower bound of the truck number *LBnTrucks* (see Section 4), it can be stated that for 115 of 126 instances (i.e. 91.3%) the optimal truck number was reached. For the residual instances, the lower bound was failed by one truck. The values of the height gap (as defined above) turn out to be moderate. Now the height gap never exceeds 31% and it amounts to 9.3% on average.

Finally, for the 12 LPIT instances of *set 4* with 180 pallets per instance, the results obtained were stable and satisfactory. The lower bound of the truck number *LBnTrucks* was missed by one truck in any case. The height gap amounts 29.0% on average.

All in all, the test by means of the representative third and fourth instance sets proves that LPIT is an effective and efficient solution procedure for loading pallets into trucks with predefined stack places. It should be stressed here again that the lower bound of the truck number is rather weak since it does not take into account the LIFO condition.

In Fig. 5, pallet charts are presented for an arbitrarily chosen instance of set 3. Each chart visualizes the loading of pallets within one strip (left or right) of one truck. Since only one truck was filled, two strips are shown. Pallets of different customers are differently shaded. Besides a pallet index (left index), each pallet has a customer position index (right index) indicating the position of the customer in the given (single) route of the instance. It is easily seen that the LIFO constraint (C7) is completely satisfied.

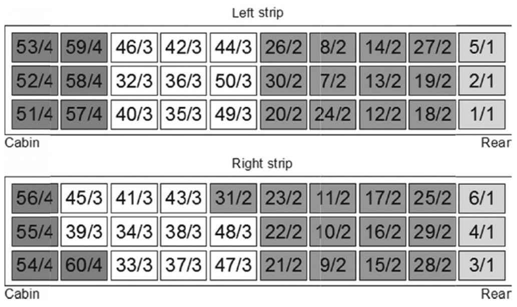

Fig. 5. Pallet charts for one loaded truck.

7. Conclusions

In this paper, a two-stage packing problem is considered that occurs in the distribution process of a Portuguese trading company. Boxes are to be packed onto identical pallets, which in turn are to be loaded into a minimal number of uniform trucks. Each truck provides a set of places for stacking pallets that are organized in two strips. Within each of the problem stages, multiple packing constraints, such as LIFO policy and support of boxes or pallets, are to be observed. A solution method was developed consisting of two procedures that cover the two problem stages. The procedure for packing boxes onto pallets was derived from the GRASP procedure for container loading by Moura and Oliveira (2005), while a new tree search procedure was devised for LPIT.

The solution method was tested by means of problem instances whose data were taken from the distribution process of the company. The method proved to be able to yield near-optimal results regarding the number of needed trucks in short running times of less than one minute. The implementation of the solution method in the distribution process would be advantageous for the company: on the one hand, suitable stowage plans including visualization are provided quickly and automatically. On the other hand, the dispatchers can make more effective packing decisions. In particular, they are able to make proposals to customers very rapidly that may help to establish a suitable balance of customer service and utilization of truck volume. All in all, the developed method could serve as a core module of an effective decision support tool.

References

Alonso, M.T., Alvarez-Valdes, R., Iori, M., Parreño, F., Tamarit, J.M., 2015. Mathematical models for multi container loading problems. Technical report submitted to Optimization-Online repository.

Alvim, A., Ribeiro, C., Glover, F., Aloise, D., 2004. A hybrid improvement heuristic for the one-dimensional bin packing problem. *Journal of Heuristics* 10, 2, 205–229.

Araya, I., Riff, M.-C., 2014. A beam search approach to the container loading problem. *Computers & OR* 43, 100–107.

Bortfeldt, A., Wäscher, G., 2013. Constraints in container loading: a state-of-the-art review. *European Journal of Operational Research* 229, 1, 1–20.

Crainic, T., Perboli, G., Tadei, R., 2008. Extreme point-based heuristics for three-dimensional bin packing. *INFORMS Journal on Computing* 20, 368–384.

Crainic, T., Perboli, G., Tadei, R., 2009. TS 2PACK: a two-level tabu search for the three-dimensional bin packing problem. *European Journal of Operational Research* 195, 744–760.

Falkenauer, E., 1996. A hybrid grouping genetic algorithm for bin packing. *Journal of Heuristics* 2, 1, 5–30.

Fanslau, T., Bortfeldt, A., 2010. A tree search algorithm for solving the container loading problem. *INFORMS Journal on Computing* 22, 222–235.

Faroe, O., Pisinger, D., Zachariasen, M., 2003. Guided local search for the three-dimensional bin packing problem. *INFORMS Journal on Computing* 15, 267–283.

Fleszar, K., Hindi, K.S., 2002. New heuristics for one-dimensional bin-packing. *Computers & OR* 29, 7, 821–839.

Gonçalves, J.F., Resende, M.G.C., 2012. A parallel multi-population biased random-key genetic algorithm for a container loading problem. *Computers & OR* 39, 2, 179–190.

Mack, D., Bortfeldt, A., 2012. A heuristic for solving large bin packing problems in two and three dimensions. *Central European Journal of Operations Research* 20, 2, 337–354.

Martello, S., Toth, P., 1990. *Knapsack Problems: Algorithms and Computer Implementations*. John Wiley & Sons, New York.

Morabito, R., Morales, S., 1998. A simple and effective recursive procedure for the manufacturer's pallet loading problem. *Journal of the Operational Research Society* 49, 8, 819–828.

Morabito, R., Morales, S., Widmer, J., 2000. Loading optimization of palletized products on trucks. *Transportation Research Part E: Logistics and Transportation Review* 36, 4, 285–296.

Moura, A., Bortfeldt, A., 2010. A two-stage packing procedure for a Portuguese trading company. Diskussionsbeitrag der Fakultät für Wirtschaftswissenschaft der FernUniversität in Hagen, Nr. 457, working paper.

Moura, A., Oliveira, J.F., 2005. A GRASP approach to the container-loading problem. *IEEE Intelligent Systems* 20, 4, 50–57.

Moura, A., Oliveira, J.F., 2009. An integrated approach to the vehicle routing and container loading problems. *Operations Research Spectrum* 31, 4, 775–800.

Parreño, F., Alvarez-Valdes, R., Oliveira, J.F., Tamarit, J.M., 2007. A maximal-space algorithm for the container loading problem. *INFORMS Journal on Computing* 20, 3, 412–422.

Parreño, F., Alvarez-Valdes, R., Oliveira, J.F., Tamarit, J.M., 2010a. Neighborhood structures for the container loading problem: a VNS implementation. *Journal of Heuristics* 16, 1, 1–22.

Parreño, F., Alvarez-Valdez, R., Oliveira, J.F., Tamarit, J.M., 2010b. A hybrid GRASP/VND algorithm for two- and three-dimensional bin packing. *Annals of Operations Research* 179, 1, 203–220.

Schoenfield, J.E., 2002. *Fast, Exact Solution of Open Bin Packing Problems without Linear Programming*. Draft, US Army Space and Missile Defense Command, Huntsville, AL.

Scholl, A., Klein, R., Jürgens, C., 1997. BISON: A fast hybrid procedure for exactly solving the one-dimensional bin packing problem. *Computers & OR* 24, 7, 627–645.

Schwerin, P., Wäscher, G., 1999. A new lower bound for the bin packing-problem and its integration into MTP and BISON. *Pesquisa Operacional* 19, 111–129.

Silva, E., Oliveira, J.F., Wäscher, G., 2014. The pallet loading problem: A review of solution methods and computational experiments. *International Transactions in Operational Research* 23, 1–2, 147–172.

Takahara, S., 2005. Loading problem in multiple containers and pallets using strategic search method. In Torra, V., et al. (eds) *Modeling Decisions for Artificial Intelligence*. Lecture Notes in Computer Science, Vol. 3558. Springer, Berlin, Heidelberg, pp. 448–456.

Wäscher, G., Haussner, H., Schumann, H., 2007. An improved typology of cutting and packing problems. *European Journal of Operational Research* 183, 3, 1109–1130.

Zachariadis, E.E., Tarantilis, C.D., Kiranoudis, C.T., 2012. The pallet packing vehicle routing problem. *Transportation Science* 46, 3, 341–358.

Zhu, W., Lim, A., 2012. A new iterative doubling Greedy–Lookahead algorithm for the single container loading problem. *European Journal of Operational Research* 222, 3, 408–417.