received: 5 January 2018
accepted: 15 June 2018

pages: 29-40

Fast truck-packing of 3D boxes

Joanna Józefowska, Grzegorz Pawlak, Erwin Pesch, Michał Morze, Dawid Kowalski

Corresponding author:

Joanna Józefowska

Poznan University of Technology,
Institute of Computing Science, Poland
e-mail: joanna.jozefowska@cs.put.poznan.pl

Grzegorz Pawlak

Poznan University of Technology,
Institute of Computing Science, Poland
e-mail: grzegorz.pawlak@cs.put.poznan.pl

Erwin Pesch

University of Siegen, Faculty III,
Institute of Information Systems,
Center for Advanced Studies in Management, Germany
e-mail: erwin.pesch@uni-siegen.de

Michał Morze

Poznan University of Technology,
Institute of Computing Science, Poland

Dawid Kowalski

Poznan University of Technology,
Institute of Computing Science, Poland

A B S T R A C T
We present formulation and heuristic solution of a container packing problem observed in a household equipment factory's sales and logistics department. The main feature of the presented MIP model is combining several types of constraints following from the considered application field. The developed best-fit heuristic is tested on the basis of a computational experiment. The obtained results show that the heuristic is capable of constructing good solutions in a very short time. Moreover, the approach allows easy adjustment to additional loading constraints.

K E Y W O R D S
container packing, 3D-packing problem, heuristics

DOI: 10.2478/emj-2018-0009

Introduction

Global economy and competition require optimisation of the logistic chains. Transportation costs constitute a significant part of overall logistic chain operational cost. There are two major features influencing the operating costs of a transportation system. These are route and vehicle load planning. Since optimal utilisation of the load space of a vehicle may bring significant savings and contribute to decreasing the pollution of the environment, it is an extensively examined research topic. In parallel, in response to the practical needs, several software systems have been developed in this area.

The container packing problem being a special case of the three-dimensional packing problem belongs to the family of cutting and packing problems. The latest and widely cited classification of cutting and packing problems was given by Wäscher et al. (2007). In this classification, the container loading problem is the three-dimensional rectangular Single Large Object Placement Problem (SLOPP).

The vehicle loading problem is usually modelled as a three-dimensional packing problem with heterogeneous items of regular shapes. The shape of an item is a parallelepiped. Two versions of the problem are considered in the literature. In the first one, the knapsack version, the container space available is fixed, and the goal is to maximise the weight of the packed items while some items may be left unpacked. In the second version, the bin-packing, the goal is to minimise the number of containers while all the items have to be packed. The problem, called the container packing problem, is well known to be NP-hard (Garey & Johnson, 1979), so the main interest is to find a good heuristic to solve it.

Although several approaches have been proposed in the literature (we present some of them in more detail in the next section), none of them considers all real-life constraints following from the product characteristics, delivery terms or other company-specific features discussed in Section 1.1. On the other hand, the commercial software available produces solutions far from optimality.

The goal of this research is to develop a fast and effective heuristic for solving the container packing problem flexible enough to produce solutions respecting various types of real-life constraints. The paper is organised as follows. In Section 1, we present the MIP formulation of the considered problem. Section 2 contains a description of the proposed heuristic algorithm. Results of computational experiments are provided in Section 3. The last section summarises the work.

The motivation for solving the practical problem comes from a factory producing household equipment. This factory produces goods such as washing machines, fridges, ovens, microwaves etc. These goods are packed into paralepipedic cartoon boxes. Client orders are serviced by the sales and logistics department. However, customers can order goods in many different ways such as by the e-mail, fax, even telephone or by a specific web application. One of the most important issues for customers is the transportation cost which depends on the container size and the optimal usage of the space in the container. Additionally, various constraints restricting the way of transporting this kind of goods are to be considered. There are constraints connected with the stability of the load in the container or the placement of boxes in the container, such as the dependency restriction that means, for example that the washing machine box cannot be placed on the kitchen oven box. Moreover not all rotation axes are possible, e.g., the fridge cannot be placed upside down. In our case, the assortment of the products is also somewhat wide, but for one transportation order, it does not exceed twenty different types of products.

Direct clients of the factory are re-sellers, supermarket chains, wholesaler, outlets and retailers placed all over the world. Clients usually do not know or do not specify the placement of goods in containers. Hence the necessity to construct a heuristic algorithm able to produce a high quality solution within time that must not exceed two minutes. Such an approach has not been studied in the literature where many models do not respect real constraints such as product stability and box dependency.

We focus on a quickly generated solution with acceptable container space usage. Input data is the order list. The seller's work is to negotiate order lists, adjust them, usually by decreasing the number of loaded goods. Sometimes, the consultation of the packing experts is necessary to deal with the transportation constraints and reduce the transportation cost paid by the customers. After being accepted by the seller, the packing list is generated. Subsequently, the packing list is transferred to the storage accompanied by the packing order and 3D visualisation of how the purchased products should be placed in the container to reduce the additional free space usage.

The selling process is considered to be changed. The customer should be able to construct the final order list using the web application allowing him to compose the packing solution for the specific container. After acceptance, the final packing list is sent to the storage. The application is directly connected to the assortment database, and an authorized customer can make an order considering its distribution in the containers and trucks. The system significantly reduces the factory's operating and personnel costs. Moreover, it allows a deeper analysis of client orders from the transportation point of view.

1. Problem formulation

1.1. Basic MIP model

Let us consider a rectangular cuboid (parallelepiped) container of dimensions (D_x, D_y, D_z) and a set of n parallelepiped items (boxes) of dimensions (d_{li}, d_{wi}, d_{hi}), $i = 1, \ldots, n$. The basic container packing problem is to find a subset of boxes and their positions in the container so that the total volume of the boxes packed is maximised. In a feasible solution, all items must be fully inside the container and no two boxes may overlap. Fasano (2008) proposed the following MIP formulation of the above problem calling it it "the basic problem". To build the MIP model, let us assume that the container is included in the positive quadrant of an orthonormal reference frame (x, y, z) with the origin O that coincides with one of the corners of the container. Moreover, the sides of the container as well as of all the boxes inside are parallel to the reference frame axes (x, y, z). The position of box i may be thus defined by giving the coordinates (c_{xi}, c_{yi}, c_{zi}) of its geometrical centre (we assume that it coincides with the mass centre) and binary variables $\delta_{\alpha\beta i}$, $\alpha \in \{l,w,h\}, \beta \in \{x,y,z\}, i \in \{1,\ldots,n\}$ where

$$\delta_{\alpha\beta i} = \begin{cases} 1 & \text{if side } \alpha \text{ of box } i \text{ is parallel (same orientation) to axis } \beta, \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Binary variables χ_i, $i = 1, \ldots, n$, indicate whether box i is packed in the container.

$$\chi_i = \begin{cases} 1 & \text{if box } i \text{ is packed,} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Although various objectives may be considered, the one used most often in logistic applications is to maximise the volume of boxes packed (3):

$$\text{maximise} \sum_{i=1}^{n} d_{li} d_{wi} d_{hi} \chi_i \tag{3}$$

The constraints considered in the basic model guarantee that each side of every box is parallel to one side of the container, all boxes packed are fully contained inside the container, and that the boxes do not intersect. These constraints are formulated as follow.

The orthogonality constraints (4) and (5) ensure that exactly one side of a packed box is parallel to each axis, and each axis is parallel to exactly one side of the packed box.

$$\sum_{\beta \in \{x,y,z\}} \delta_{\alpha\beta i} = \chi_i, \alpha \in \{l,w,h\}, i = 1,\ldots,n \tag{4}$$

$$\sum_{\alpha \in \{l,w,h\}} \delta_{\alpha\beta i} = \chi_i, \beta \in \{x,y,z\}, i = 1,\ldots,n \tag{5}$$

The domain constraints (6) must hold for each $\beta \in \{x,y,z\}$ to guarantee that each packed box is fully contained in the container.

$$0 \le c_{\beta i} - \frac{1}{2} \sum_{\alpha \in \{l,w,h\}} d_{\alpha i}\delta_{\alpha\beta i} \le c_{\beta i} + \frac{1}{2} \sum_{\alpha \in \{l,w,h\}} d_{\alpha i}\delta_{\alpha\beta i} \le \chi_i D_\beta \tag{6}$$

Finally, the non-intersection constraints (7-9) imply that any two boxes packed do not intersect. They hold for $\beta \in \{x,y,z\}, i,j \in \{1,\ldots,n\}, i<j$.

$$c_{\beta i} - c_{\beta j} \ge \frac{1}{2} \sum_{\alpha \in \{l,w,h\}} (d_{\alpha i}\delta_{\alpha\beta i} + d_{\alpha j}\delta_{\alpha\beta j}) - (1 - \sigma^{+}_{\beta ij}) D_\beta \tag{7}$$

The binary variable $\sigma^{+}_{\beta ij}$ is introduced to detect the situation where boxes i and j are packed in the container, they do not intersect along axis β, and j precedes i, (i.e. $c_{\beta j} < c_{\beta j}$). If this is not the case ($\sigma^{+}_{\beta ij} = 0$), the constraint (7) is trivially satisfied. If i precedes j, the corresponding non-intersection constraints are formulated as follow:

$$c_{\beta j} - c_{\beta i} \ge \frac{1}{2} \sum_{\alpha \in \{l,w,h\}} (d_{\alpha i}\delta_{\alpha\beta i} + d_{\alpha j}\delta_{\alpha\beta j}) - (1 - \sigma^{-}_{\beta ij}) D_\beta \tag{8}$$

The binary variable $\sigma^{-}_{\beta ij}$ has a similar interpretation as $\sigma^{+}_{\beta ij}$. To guarantee that if both i and j are packed at least one of the non-intersection constraints holds, we introduce constraints (9).

$$\sum_{\beta \in \{x,y,z\}} (\sigma^{+}_{\beta ij} + \sigma^{-}_{\beta ij}) \ge \chi_i + \chi_j - 1, i,j \in \{1,\ldots,n\}, i<j \tag{9}$$

Other MIP models of the basic container packing problem were given by Onodera et al. (1991), Chen et al. (1995), Padberg (1999), Fasano (1999), Fasano (2004), Fasano (2008), Pisinger and Sigurd (2005).

1.2. Additional constraints

Many authors conclude that the basic model does not consider numerous constraints that occur in real-life container loading problems. Those constraints may be caused by the product characteristics (e.g. nothing else may be packed on top of the product because it is fragile), the loading process (by fully automated loading, some additional space is required for the manoeuvres of the loading equipment), vehicle carrying the container (different requirements for car and rail containers) etc. Although this need is widely recognised, only few heuristic approaches address the above issues explicitly.

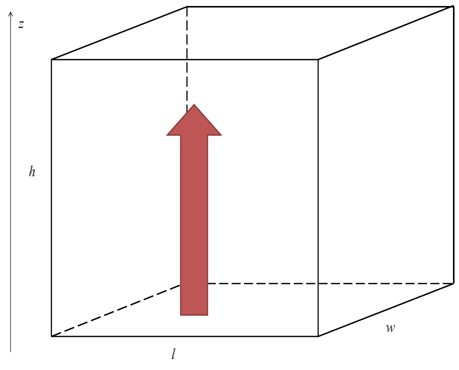

Fig. 1. Box orientation

The first systematic survey of practically justified requirements affecting the container load was presented by Bischoff and Ratcliff (1995). Below, we give a short survey of additional constraints related to the allocation of boxes or box size, the mass of the boxes and load stability.

1.2.1. Volumetric constraints

Volumetric constraints are generally considered by the basic formulation. The sizes of the container and boxes are basic data for the loading problem. It is often the case that a box must be positioned with exactly one side at the top as shown in Fig. 1. It is easy to model fixed (or partially fixed) orientation of boxes. To fix the orientation of box *i*, it is enough to fix the values of the corresponding $\delta_{\alpha\beta i}$ variables. We may conclude that all MIP formulations and most of the construction heuristics allow fixing the orientation of boxes and so does our method as well.

1.2.2. Mass constraints

One of the very important constraints sometimes considered as an objective is related to the maximum mass of the loaded items (10). In some situations, the maximum load of the carrying vehicle may be limited. It is the case for tucks since in many countries the maximum load is regulated by law. Such a constraint may be naturally modelled by adding the following constraint:

$$\sum_{i=1}^{n} m_i \chi_i \leq M, \tag{10}$$

where m_i is the mass of box *i*, $i = 1, \ldots, n$, and *M* is the maximum load of the container.

Moreover, in some applications (aircraft loading or a container moved by a crane), the distribution of mass is a crucial issue. Another limitation may follow from the bearing strength of a box and may be expressed as the maximum weight of boxes packed on top of a given box.

1.2.3. Layer constraints

In many practical situations, loading constraints require that some boxes have to be placed on the floor, or nothing else may be placed on top of some fragile boxes. In general, some relation may be defined on the set of boxes showing if box *i* may be placed on top of box *j*. We represent this relation by a directed graph *G*, where arc (*i*, *j*) belongs to the graph if box *i* cannot be placed directly on top of box *j*. This relation is not necessarily transitive and may be symmetric. In our mathematical model, parameter g_{ij} is equal to 1 if arc (*i*, *j*) belongs to graph *G* and zero otherwise.

We illustrate the concept of graph *G* in Fig. 2. Arc (2, 3) indicates, e.g., that box 2 cannot be placed on top of box 3. Thus, the boxes cannot be packed as shown in the left scheme.

We define the variables $\gamma_{ij} = 1$ if boxes *i* and *j* are in the container, $c_{zi} > c_{zj}$ and $c_{zi} - c_{zj} - \frac{1}{2}\sum_{\alpha\in\{l,w,h\}}(d_{\alpha i}\delta_{\alpha zi} + d_{\alpha j}\delta_{\alpha zj}) = 0$ and zero otherwise, $i \neq j$.

Let us first ensure that if both boxes *i* and *j* are packed then $\gamma_{ij} = 1$:

$$\gamma_{ij} \geq \chi_i + \chi_j - 1, i \neq j \tag{11}$$

and that if any of the boxes *i*, j are not packed then $\gamma_{ij} = 0$:

$$2 * \gamma_{ij} \leq \chi_i + \chi_j, i \neq j \tag{12}$$

The following equation shows that if $\gamma_{ij} = 1$ then the distance between the coordinates of boxes *i* and *j* along axis *z* (the vertical one) allows that box *i* is exactly on top of box *j*.

$$\gamma_{ij}\left[c_{zi} - c_{zj} - \frac{1}{2}\sum_{\alpha\in\{l,w,h\}}(d_{\alpha i}\delta_{\alpha zi} + d_{\alpha j}\delta_{\alpha zj})\right] = 0 \tag{13}$$

Finally, we formulate the condition (14) making sure that if $\gamma_{ij} = 1$ and $g_{ij} = 1$ (box *i* cannot be placed on top of box *j*) then the boxes do not intersect along at least one axis *X* or *Y*.

$$\sigma^+_{xij} + \sigma^-_{xij} + \sigma^+_{yij} + \sigma^-_{yij} \geq g_{ij}\gamma_{ij} \tag{14}$$

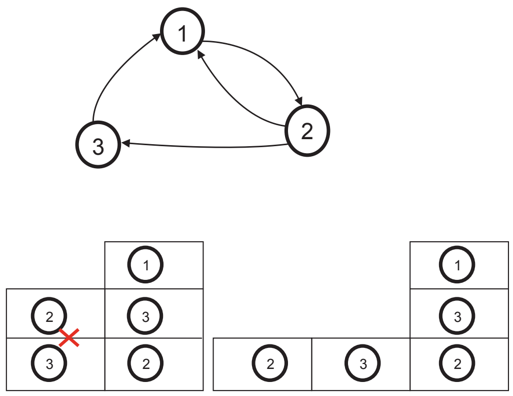

Fig. 2. Part dependency graph and the example solution

1.2.4. Load stability constraints

Load stability constraints are a combination of spatial and mass constraints. The main idea is to allocate the boxes in such a way that they do not fall not only in a static position but also while the container is being moved. The most challenging situations occur when it is being turned and lifted or when the vehicle carrying the container speeds up or slows down rapidly.

Bischoff and Ratcliff (1995) propose a heuristic solution that produces patterns which combine efficient space utilisation with a high degree of stability. The algorithm generating stable loads is based on earlier work by Bischoff (1991) on stability aspects of pallet loading. Quite an efficient algorithm combining a construction heuristic with a search algorithm for the problem with limited load bearing strength of boxes was proposed by Bischoff (2006).

For three-dimensional problems Fasano (2004; 2008) expressed the requirement that the overall centre of gravity of a container must stay within a given convex domain. Also, for three-dimensional problems Takadama et al. (2004) and Egeblad (2009) cope with this balancing requirement in combination with the objective function. Egeblad additionally includes inertia moment characteristics of the loaded container. De Castro et al. (2003) and Junqueira et al. (2012) look upon stability, bearing and dropping constraints too.

Amiouny et al. (1992), and Mathur (1998) considered a slightly different problem called "balanced loading" where the goal is to generate a pattern for which the centre of gravity is as close to a predefined point as possible. The motivation for such a goal comes from aircraft loading. Both papers present heuristics for a one-dimensional problem.

We assume that the load is stable if the gravity is compensated by the support of another box or the container floor. We calculate the stability using the centre of mass of a box allocated over another one. We also assume that the centre of mass coincides with the geometrical centre of the box which is acceptable in most practical situations.

We introduce variables $\lambda_{ij} = 1$, $i \neq j$ if boxes i and j are packed and box j supports box i and zero otherwise. Moreover, we assume that $\lambda_{i0} = 1$ if i is packed and placed on the floor of the container and zero otherwise.

Constraint (15) ensures that if the distance between the centres of box i and j in the vertical axis is not exactly equal to the half-size of the corresponding sides then box j may not support box i (and vice versa). It also ensures that $\lambda_{ij} = 0$ if any of the boxes i or j are not packed.

$$\lambda_{ij} \leq \gamma_{ij}, i, j \in \{1,\ldots,n\}, i \neq j \tag{15}$$

The next equation ensures that box i may be placed on the floor only if box i is packed in the container:

$$\lambda_{i0} \leq \chi_i, i = 1,\ldots,n \tag{16}$$

Inequalities (17) and (18) hold if in case j supports i, the projection of the centre of mass of box i on the plane (X, Y) fits inside the rectangle being the projection of box j.

$$\lambda_{ij}\left[c_{xi} - c_{xj} - \frac{1}{2}\sum_{\alpha\in\{l,w,h\}} d_{\alpha j}\delta_{\alpha xj}\right] \leq 0, i, j \in \{1,\ldots,n\}, i \neq j \tag{17}$$

$$\lambda_{ij}\left[c_{yi} - c_{yj} - \frac{1}{2}\sum_{\alpha\in\{l,w,h\}} d_{\alpha j}\delta_{\alpha yj}\right] \leq 0, i, j \in \{1,\ldots,n\}, i \neq j \tag{18}$$

Equation (19) states that for the boxes located on the floor of the container the coordinate of the centre of mass along axis z is exactly one half of the length of the corresponding side of the box.

$$\lambda_{i0}\left(c_{zi} - \frac{1}{2}\sum_{\alpha\in\{l,w,h\}} d_{\alpha i}\delta_{\alpha zi}\right) = 0, i = 1,\ldots,n \tag{19}$$

Finally, equation (20) guarantees that each box is either packed on the floor or is supported by another box.

$$\sum_{j=0, j\neq i}^{n} \lambda_{ij} \geq 1, i = 1,\ldots,n \tag{20}$$

The heuristic algorithm presented in Section 2 also takes into account the stability constraints.

1.2.5. Other constraints

Other constraints may follow from the organisational reasons, for example, when the load has many destinations, and some items have to be unloaded earlier and others later at another destination. Then, it is convenient to allocate the items according to the unloading order.

Bischoff and Ratcliff (1995) proposed a heuristic that considered the multi-drop situation and constructs a pattern with distinct sections across the width of the container which correspond to the different destinations.

We propose to model these constraints by dividing the space of the container into "subcontainers" corresponding to consecutive drops and pack each subcontainer separately.

2. Solution method

2.1. Related work

The first and somewhat sophisticated heuristics for the container packing problem was proposed by George and Robinson (1980). It was a wall-building procedure that was later improved and verified for various ranking rules by Bischoff and Mariott (1990). Also, Pisinger (2002) constructed an algorithm using the concept of wallbuilding. His heuristic decomposes the problem into some layers which again are split into some strips. The packing of a strip may be formulated and solved optimally as a Knapsack Problem with a capacity equal to the width or height of the container. The depth of a layer as well as the thickness of each strip is decided through a branch-and-bound approach where at each node only a subset of branches is explored. Eley (2002) proposed an algorithm being a combination of a greedy heuristic generating blocks of boxes with a tree search procedure. Another approach using a slightly modified concept of wall building with a greedy randomised adaptive search procedure (GRASP) was examined by Moura and Oliveira (2005) and Parreno et al. (2010).

Other approaches are presented in the literature where one can find a 3D-BPP problem in Sciomachen (2007) (items are containers, and the bins are ships). Also, in Terno et al. (2000) the multi-palleting 3D loading problem with the minimal number of pallets is considered.

Several metaheuristic approaches to solving the container packing problem have been proposed as well starting with the genetic algorithm by Gehring and Bortfeldt (1997). In their further works, the same authors developed a tabu search algorithm (Bortfeldt & Gehring, 1998), a hybrid genetic algorithm (Bortfeldt & Gehring, 2001) and a parallel genetic algorithm (Gehring & Bortfeldt, 2002). A parallel tabu search algorithm (Bortfeldt et al., 2003) and a hybrid local search algorithm combining simulated annealing and tabu search (Mack et al., 2004) were developed in continuation of the earlier research.

2.2. Algorithm

Numerous heuristic approaches have been proposed in the literature to solve the basic container loading problem. Only a few of them consider additional constraints described in Section 1. We propose the best fit heuristic based on the idea of wall building that in addition to volumetric and mass constraints considers layer and load stability constraints. In particular, the following constraints are implemented:

- orientation of selected boxes may be arbitrarily fixed,
- layer constraints described in Section 1.2.3 may be defined by a corresponding directed graph,
- load stability constraints described in Section 1.2.4 are respected.

Basically, the packing scheme follows the wall-building idea by George and Robinson (1980). The container is packed layer by layer, where a layer is defined as a section of the container length over its total width and height. The length of a layer is defined by the first box packed in this layer. No new layer may be initiated unless the previous layer is completed.

We define *available space* as a parallelepiped inside the container with sides parallel to the sides of the container and neither intersecting nor containing any box. An available space S^i_j may be divided by a new box i+1 as shown in Fig. 3. This operation results in removing the available space S^i_j, adding box i+1 to the container and constructing a set of available spaces $S^{i+1}_{j1}, \dots, S^{i+1}_{jk}$ so that:

(a) $S^{i+1}_{jl} \subseteq S^i_j, l = 1, \dots, k,$

(b) S^{i+1}_{jl} is disjoint with box i+1, $l = 1, \dots, k,$

(c) $\bigcup_{l=1}^{k} S^{i+1}_{jl}$ has maximum volume of all sets that fulfil (a) and (b).

To decrease the running time of the algorithm, if available space, is a subset of another available space it is removed from further consideration.

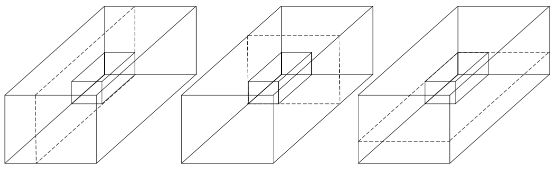
Fig. 3. Available spaces resulting from packing a box

Let us observe that if no orientation constraints exit, six different orientations of a box are possible as shown in Fig. 4.

The heuristic tries to pack a box in every feasible orientation until it fits into available space. If no such space exists for any orientation, the box is rejected. It is easy to notice that the order in which possible orientations are checked may influence the solution. Since we have six possible orientations at the most, there exist only 6! = 720 such orders (permutations). We run the heuristics for each permutation.

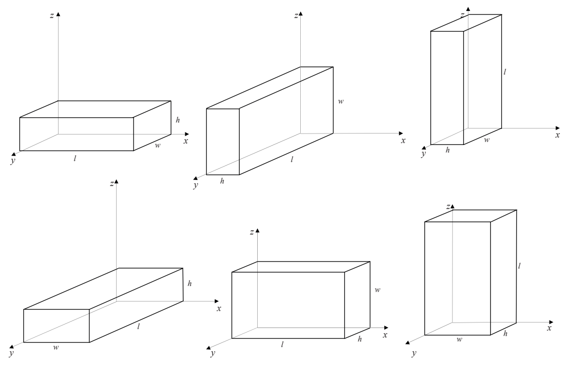

Fig. 4. Different box orientations

Thus, in a given run of the algorithm, the order in which box orientations are checked is fixed. Boxes are considered in the non-increasing order of their volume. If there is no available space where the box may be packed in a given orientation, then the next orientation is selected. If a box cannot be packed in any orientation, it is rejected together with all boxes of the same type. Then, the next box is selected. After considering all boxes, the next run of the algorithm starts with a new permutation of box orientations. The best solution of all 720 runs is selected.

A more formal description of the algorithm is presented in Section 2.3.

2.3. The Best-Fit algorithm

For every possible permutation of box orientations do:

1. Add the initially available space corresponding to the whole container to the set of available spaces.
2. Select the largest box not yet considered. If there are no boxes to be packed, go to step 5.
3. For every orientation from the current permutation do:
   a) select the smallest yet unchecked available space and try to pack the current box there,
   b) if all constraints are met, then:
      - pack that box respecting its current orientation,
      - divide all available spaces intersecting with the selected box and add the new available spaces to the set of available spaces,
      - remove each available space that is included in another available space,
      - go to step (2).
   c) if there are still unchecked spaces, go to step (3a).
4. Reject the selected item and all items of its type. Go to step 2.
5. If the current solution is better than all of the solutions found so far, then remember it.
6. Remove the current solution.

If no box has been rejected then all boxes are packed; otherwise, there are some unpacked boxes left. Step 4 is not required to obtain a feasible solution; however, it decreases the complexity of the algorithm. Otherwise, the number of available spaces grows very fast.

Obviously, more complicated orientation requirements may be defined by the user. For example, additional space required for manoeuvring of the loading vehicles (forklifts or pallet trucks) may be required. This additional space obviously decreases the container volume available for packing the boxes thus affecting the value of the objective function. Moreover, the location of the door may influence the order of packing and in consequence the location of some items. Such constraints may be considered by the proposed heuristics by defining the available space appropriately.

3. Computational experiments

3.1. Test-bed

The algorithm was implemented in Java, and the tests were run on a computer with Intel Core 2

Quad CPU Q9650 with 3.00GHz, 8GB of core memory and OpenSuSE Linux as a single-threaded application with 120 seconds of processing time.

The set of test instances most often used in the literature was proposed by Bischoff and Ratcliff (1995). In this set, however, no additional constraints like orientation, load stability or layers constraints were defined. Nevertheless, our first experiments were performed on this set of instances with 3, 5, 8, 10, 12, 15 and 20 boxes. The results were on average no more than 9% worse in terms of the volume utilisation as compared to the algorithms by Bischoff and Ratcliff (1995); Bischoff (2006); Lim et al. (2005). Only the GRASP algorithm by Parreno et al. (2010) showed significant advantage (more than 10% on average) but at the expense of longer computation. These results were promising enough to continue the experiments on instances with additional constraints. Although we cannot compare our algorithm with other algorithms, the obtained results have been assessed by practitioners as highly satisfactory in a real setting. There is also a natural upper bound of volume utilisation which is 100% because it follows directly from the problem formulation, so the user has quite precise information about the quality of the solution even without knowing the optimal solution.

Although the motivation for our model comes from a practical application, unfortunately, no real order test instances are available. To evaluate the algorithm performance, we have prepared a set of instances based on a list of real products. The real data include box size, mass and the possible box orientation. The list contains 99 types of products. An instance is constructed by randomly choosing boxes from the list to set up a shipment order.

The experiment was performed on a set of 720 instances where ten instances were randomly generated from each of 72 groups. The groups of instances were defined by providing:

- container size (3),
- number of box types (4),
- number of pivot points (2),
- volume ratio (3).

The total number of combinations of the above parameters gives 72 instance groups.

The first typical container size was assumed to be 2.4 m wide, 2.6 m high and 12.2 m long. Two other types were two and three times shorter, respectively.

The number of different box types is assumed to be 2, 5, 10 or 20 chosen from the list of 99 different box types. The following algorithm was proposed to select box types constituting an instance:

- order all box types by non-decreasing volume,
- define a range of items as items at most p positions from a pivot value point,
- select box types evenly from the defined range.

There could be one or two randomly chosen pivot value points. We assumed $p = 0.3$ for a single point and $p = 0.5$ in case of two preference points (Fig. 5).

Finally, the number of boxes of each type is decided as follows. In each instance, a group of "smaller" and "bigger" boxes are distinguished based on the volume of the selected box types. Depending on the number of box types in an instance we have:

- two box types: two groups of one item each,
- five box types: the first group of two items, the second group of three items,

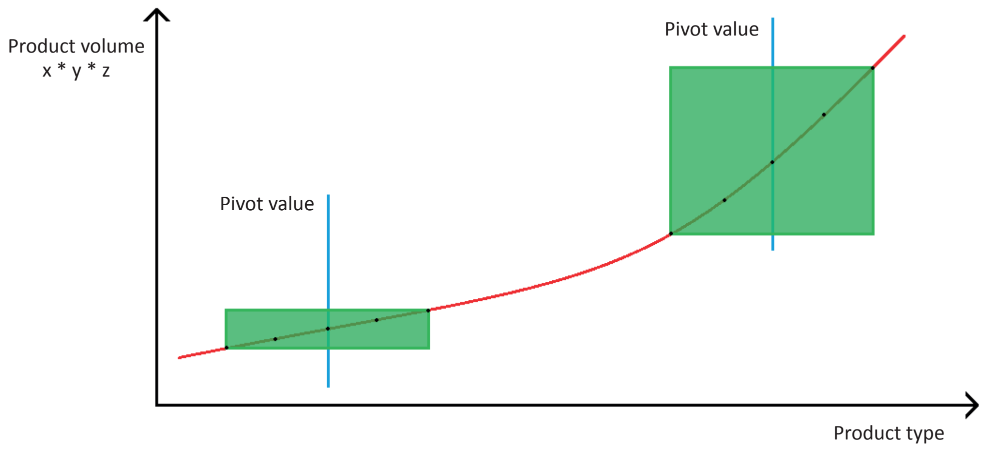

Fig. 5. Generating box size of the volume distribution for two pivot values

- ten box types: two groups of five items each,
- twenty box types: two groups of ten items each.

The total volume of all boxes in an instance is a random number between 110% and 150% of the container capacity volume. Next, generated boxes are divided into two groups. The volume ratio (50%–50%; 20%–80% – more items from the second group; and 80%–20% – more items from the first group), allows to calculate the approximate total volume of items, from the first and second group, respectively. Finally, for each box type in the group, the number of items is obtained from a uniform distribution so that the total volume of the group is reached.

Eventually, for every instance, the graph of layer constraints (Fig. 2) is randomly generated.

3.2. Results

The computational results are summarised in Tab. 1. The quality of the solution and computational execution time for different container capacity, from *C*1 to *C*3 and different type of boxes, from 2 to 20, are presented in Fig. 6 and Fig. 7. The main entries in Tab. 1 give the usage of container capacity in percentage.

The algorithm presented in the previous section is tested on a set of instances representing the data for the real products and the real size of the containers. It was also necessary to test algorithms for the smaller containers to reflect the situation where a container is divided into two or three spaces because the product is delivered to two or three different customers.

We obtain comparative results through implementation of our MIP model in CPLEX 12.3. CPLEX solves only the very simple and unrealistic instances of several boxes in the time of 2 minutes. No instance from the listing (Tab. 1) could be solved within a 10 hours time limit, and our MIP-based approach was not competitive and useful in the practical case.

In Fig. 6, the percentage distance (in volume usage) from the potentially possible 100% solution is shown. One can observe that the solutions of capacity usage delivered by the algorithm depend on the number of different types of boxes. For example, for two types of boxes, the simplest case, the utilisation of available volume space in the container is more 90%, on average. The value of the results decreases when the number of types of boxes increases. In the practical situations, two to five types of boxes are packed on the truck.

For these cases, the average usage of capacity is over 80% which is a generally expected result in practice. However, in particular cases, the space usage in the truck could be much lower, especially when the dependency graph is very restricted, for example in the case, where only one layer of the product is allowed.

Fig. 7 depicts execution times over all original instances. The plots in the Fig. 7 show minimum, first, second, third quartile, and maximum execution times. There is an upper limit of 120 seconds of the execution time. The execution time of such kind of algorithms naturally depends on the container size

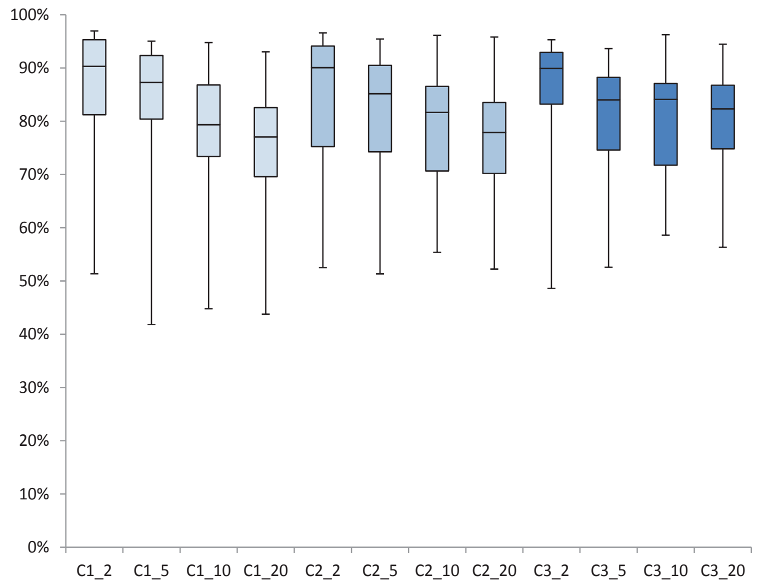

Fig. 6. Volume usage for three sizes of containers and different numbers of box types

Volume 10 • Issue 2 • 2018

Tab. 1. Results for three different container sizes

	Ratio	Pivot value	Number of types			
			2	5	10	20
Container 1 (40 ft)	50/50	1	82.04	82.24	72.47	76.01
		2	92.92	92.40	80.22	78.92
	20/80	1	80.39	80.78	83.07	69.92
		2	96.54	92.16	81.86	83.55
	80/20	1	77.67	71.13	73.50	65.48
		2	90.17	85.50	76.73	81.02
	Ratio	**Pivot value**	**Number of types**			
			2	**5**	**10**	**20**
Container 2 (20 ft)	50/50	1	80.86	73.40	76.84	74.08
		2	94.10	87.27	80.00	76.47
	20/80	1	83.40	80.93	74.47	78.73
		2	96.60	86.91	81.97	81.53
	80/20	1	69.95	71.00	74.23	68.19
		2	79.46	84.30	82.83	82.04
	Ratio	**Pivot value**	**Number of types**			
			2	**5**	**10**	**20**
Container 3 (13 ft)	50/50	1	80.28	79.34	76.08	81.82
		2	91.49	88.50	85.58	78.19
	20/80	1	79.14	73.12	85.49	77.62
		2	93.55	85.06	83.03	83.35
	80/20	1	76.35	73.93	75.07	77.63
		2	86.34	87.03	76.64	82.01

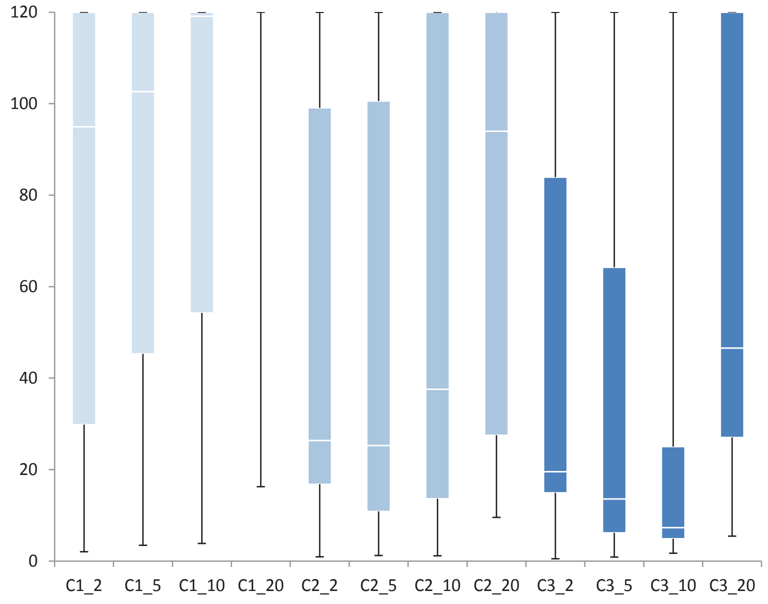

Fig. 7. Execution time of the algorithm for three sizes of containers and different numbers of box types

and the number of types of boxes, which is depicted in Fig. 7.

Overall, it can be concluded that the heuristic provides an 80% capacity usage solution within the imposed execution time. This result is satisfactory from the application of the methods point of view.

Conclusions

In this paper, we propose a MIP model and the solution of a 3D packing problem. Though this problem has similarities with other well known OR and packing problems, it is fundamentally different because we consider a real shipment environment and a part of a real application is in the focus of our solution approach. Generally, this problem is computationally hard, but the results obtained by a heuristic proved to be practically "good enough" to generate packings that have been highly appreciated by the company.

Literature

Amiouny, S. V., Bartholdi, J. J. I., Vande Vate, J. H., & Zhang, J. (1992). Balanced loading. *Operations Research*, *40*, 238-246.

Bischoff, E. (1991). Stability aspects of pallet loading. *OR Spectrum*, *13*,189-197.

Bischoff, E. (2006). Three dimensional packing of items with limited load bearing strength. *European Journal of Operational Research*, *168*, 952-966.

Bischoff, E., & Mariott, M. (1990). A comparative evaluation of heuristics for container loading. *European Journal of Operational Research*, *44*, 267-276.

Bischoff, E., & Ratcliff, M. (1995). Issues in the development of approaches to container loading. *OMEGA*, *23*(4), 377-390.

Bortfeldt, A., & Gehring, H. (1998). A tabu search algorithm for weakly heterogeneous container loading problems. *OR Spektrum*, *20*, 237-250.

Bortfeldt, A., & Gehring, H. (2001). A hybrid generic algorithm for the container loading problem. *European Journal of Operational Research*, *131*, 143-161.

Bortfeldt, A., Gehring, H., & Mack, D. (2003). A parallel tabu search algorithm for solving the container packing problem. *Parallel computing*, *29*, 641-662.

Chen, C., Lee, S., & Shen, Q. S. (1995). An analytical model for the container loading problem. *European Journal of Operational Research*, *80*, 68-76.

De Castro, J., Silva, N., Soma, N., & Maculan, N. (2003). A greedy search for the three-dimensional bin packing problem: the packing static stability case. *International Transactions in Operational Research*, *10*, 141-153.

Egeblad, J. (2009). Placement of two-and three-dimensional irregular shapes for inertia moment and balance. *International Transactions in Operational Research*, *16*(6), 789-807.

Eley, M. (2002). Solving container loading problems by block arrangements. *European Journal of Operational Research*, *141*, 393-409.

Fasano, G. (1999). Cargo analytical integration in space engineering: A three dimensional packing model. In T. Ciriani, S. Gliozzi, E. Johnson (Eds.), *Operations Research in Industry* (pp. 232-246). London, England: Macmillan.

Fasano, G. (2004). A MIP approach for some practical packing problems: Balancing constraints and tetris-like items. *Quarterly Journal of Operations Research*, *2*(2), 161-174.

Fasano, G. (2008). MIP-based heuristic for non-standard 3D-packing problems. *Quarterly Journal of Operations Research*, *6*(3), 291-310.

Garey, M., & Johnson, D. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco, USA: W. H. Freeman and Company.

Gehring, H., & Bortfeldt, A. (1997). A genetic algorithm for solving the container loading problem. *International Transactions in Operational Research*, *4*(5/6), 401-418.

Gehring, H., & Bortfeldt, A. (2002). A parallel genetic algorithm for solving the container loading problem. *International Transactions in Operational Research*, *9*, 497-511.

George, J. A., & Robinson, D. F. (1980). A heuristic for packing boxes into a container. *Computers and Operations Research*, *7*, 147-156.

Junqueira, L., Morabito, R., Yamashita, D., & Yanasse, H. (2012). *Optimization Models for the Three-dimensional Container Loading Problem with Practical Constraints*. New York, USA: Springer Science, & Business Media.

Lim, A., & Rodrigues, B. (2005). 3-D container packing heuristics. *Applied Intelligence*, *22*(2), 125-134.

Mack, D., Bortfeldt, A., & Gehring, H. (2004). A parallel local algorithm for the container loading problem. *International Transactions in Operational Research*, *11*, 511-533.

Mathur, K. (1998). An integer-programming-based heuristic for the balanced loading problem. *Operations Research Letters*, *22*, 19-25.

Moura, A., & Oliveira, J. (2005). A GRASP approach to the container-loading problem. *IEEE Intelligent Systems*, *20*, 50-57.

Onodera, H., Taniguchi, Y., & Tamaru, K. (1991). Branch-and-bound placement for building block layout. *28th ACM/IEEE design automation conference*, 433-439.

Padberg, M. (1999). Packing small boxes into a big box. *Mathematical Methods of Operations Research*, *52*(1), 1-21.

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

Pisinger, D. (2002). Heuristics for the container loading problem. *European Journal of Operational Research*, *141*, 382-392.

Pisinger, D., & Sigurd, M. (2005). The two-dimensional bin packing problem with variable bin sizes and costs. *Disctrete Optimization*, *2*(2), 154-167.

Sciomachen, A., & Tanfani, E. (2007). A 3D-BPP approach for optimising stowage plans and terminal productivity. *European Journal of Operational Research*, *183*(3), 1433-1446.

Takadama, K., Tokunaga, F., & Shimohara, K. (2004). Capabilities of a multiagent-based cargo layout system for h-ii transfer vehicle. *16th IFAC Symposium on Automatic Control in Aerospace*, 250-255.

Terno, J., Scheithauer, G., Sommerweiss, U., & Riehme, J. (2000). An efficient approach for the multi-pallet loading problem. *European Journal of Operational Research*, *123*, 372-381.

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