versão impressa ISSN 0101-7438 / versão online ISSN 1678-5142

A MULTI-START RANDOM CONSTRUCTIVE HEURISTIC FOR THE CONTAINER LOADING PROBLEM

Olinto César Bassi de Araújo*
Colégio Politécnico da Univ. Federal de Santa Maria
Universidade Federal de Santa Maria (UFSM)
olinto@smail.ufsm.br

Vinícius Amaral Armentano
Faculdade de Engenharia Elétrica e de Computação
Universidade Estadual de Campinas (UNICAMP)
vinicius@densis.fee.unicamp.br

* *Corresponding author* / autor para quem as correspondências devem ser encaminhadas

Recebido em 07/2006; aceito em 02/2007
Received July 2006; accepted February 2007

Abstract

This paper deals with the container loading problem which involves the selection of a subset of boxes, each box with a given volume, such that they fit in a single container and maximize its volume utilization subject to orientation and stability constraints. We propose a multi-start random constructive heuristic with a load arrangement that is based on maximal cuboids that fit in given empty spaces. Each instance is adaptively evaluated by a set of criteria, and at each step of the construction process one maximal cuboid is chosen probabilistically from a restricted list of candidates. In order to enhance the flexibility in the construction of a solution, a probabilistic reduction on such cuboids is allowed. Computational tests on several instances from the literature show that the proposed method performs better than other approaches.

Keywords: container loading; cuboid arrangement; multi-start random constructive heuristic.

Resumo

Neste trabalho abordamos o problema de carregamento de contêiner que trata da seleção de um subconjunto de caixas, cada caixa com um dado volume, de forma a maximizar o volume ocupado de um único contêiner sujeito a restrições de orientação e estabilidade. Propomos uma heurística construtiva aleatória com múltiplos inícios que utiliza um arranjo de carga baseado em cubóides que maximizam a ocupação de espaços vazios. Cada instância é avaliada de forma adaptativa por um conjunto de critérios, e em cada passo do processo construtivo um cubóide é selecionado probabilisticamente de uma lista restrita de candidatos. Para aumentar a flexibilidade na construção de uma solução, permite-se uma redução probabilística no tamanho dos cubóides. Resultados computacionais em instâncias da literatura mostram que o método proposto apresenta um desempenho superior a outros enfoques sugeridos na literatura.

Palavras-chave: carregamento de contêiner; arranjo com cubóides; heurística construtiva aleatória com múltiplos inícios.

1. Introduction

This paper addresses the problem of optimizing the loading of rectangular boxes of different sizes into a rectangular container of given dimensions so that their edges lie parallel to the edges of the container and no two items overlap. This is one of the problems with numerous applications in the cutting and packing industry and a general classification of such problems is provided by Dyckhoff (1990). The problem is particularly important in companies whose logistic activities involve storage, distribution and/or collection of goods, for a better space utilization allows reduction of cost and time in loading and unloading containers.

Three-dimensional cutting and packing problems are extensions of their one-dimensional counterparts, and therefore belong to the NP-hard class. This implies that most likely optimal methods are not able solve real problems in a reasonable time, and for this reason the literature on optimal methods and approximate algorithms is scarce, whereas the literature on heuristic methods is fairly vast.

Chen *et al.* (1995) propose a mixed integer linear programming for the problem of loading multiple containers. In addition to constraints that avoid overlapping, constraints that control the weight imbalance along one of the dimensions are also modeled. The model is tested on one instance with one container and six boxes. Lai *et al.* (1998) propose a graph-based model for the loading problem with multiple costumers orders such that cargoes belonging to the same costumer are packed together in the container. An exact algorithm and a heuristic are proposed for solving the model. Martello *et al.* (2000) propose a branch-and-bound algorithm for loading a single container, which is then used in an exact algorithm for the three-dimensional bin packing. Hifi *et al.* (2004) suggest two optimal algorithms for solving unconstrained three-dimensional cutting problems, in which there is an unlimited quantity of pieces of each type to be cut.

Heuristic approaches represent viable alternatives to obtain good solutions for practical problems in a reasonable time. Pisinger (2002) classifies heuristic approaches according to the loading building pattern, namely wall building, stack building, guillotine cutting, and cuboid arrangement.

The wall building approach constructs vertical or horizontal layers which reduce the solution space and allows the use of simple data structure in the implementation of algorithms. Such an approach was introduced by George & Robinson (1980) who suggested a sophisticated constructive heuristic based on vertical layers such that spaces not occupied in a layer can be used in subsequent layers. The ideas proposed by George & Robinson (1980) are the base for heuristics developed by Bischoff & Marriot (1990), Gehring *et al.* (1990), Bortfeldt & Gehring (2001), Pisinger (2002), Cecílio & Morabito (2004) and Moura & Oliveira (2005). Bischoff *et al.* (1995) and Lim & Zhang (2005) use horizontal layers in order to build the loading pattern.

The stack building approach allows the decomposition of the original problem into two subproblems: the three-dimensional problem of packing the boxes into suitable stacks and the two-dimensional problem of locating the stacks at the floor of the container. A stack is built from the selection of a base box that is positioned at the floor of the container. The next box that is placed in the stack must have its base fully supported by one or more the boxes that lie below it, as shown in Figure 1. This approach favors the treatment of weight constraints, but in general gives rise to loading patterns with poor horizontal stability, and when the cargo is weakly heterogeneous it results in a low utilization of the container space.

The use of this approach can be found in the heuristics proposed by Haessler & Talbot (1990), Gehring & Bortfeldt (1997).

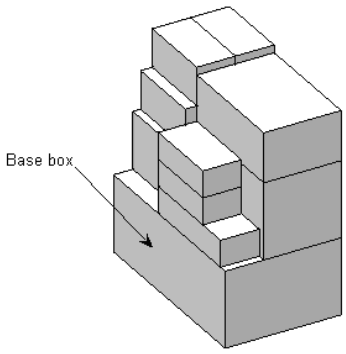


Figure 1 – Example of stack packing.

A guillotine cut is a constraint imposed to the problem, i.e., each cut on any parallelepiped produces two new parallelepipeds. In the guillotine cutting approach all boxes can be distinguished as a sequence of guillotine cuts. Hassamontr (2003) develops a software tool for three-dimensional cuts in a wooden furniture industry. Morabito & Arenales (1994) propose a heuristic which makes use of guillotine cuts as a strategy to obtain competitive results compared to non-guillotine cuts.

In the cuboid arrangement, the container is filled by homogeneous blocks made up of boxes of the same type and with identical orientation. Bortfeld & Gehring (1998) propose a heuristic that makes use of local arrangements with one or two blocks. A parallel version of this heuristic is suggested in (Bortfeldt *et al.*, 2003). Eley (2002) points out that the ease of arrangement and lower complexity in terms of load bearing strength are some of the advantages of this approach.

Ngoi *et al.* (1994) develop a spatial representation technique to model the loading process which allows the evaluation of all potential placement locations. Such a representation is used by Bischoff & Ratcliff (1995), Chien & Deng (2004) and an adaptation of the representation is utilized by Bischoff (2006). Lim *et al.* (2003) propose a heuristic in which the walls of the container are used as the ground to build up a load arrangement, called multi-faced buildup.

Real life container loading problems are complex and usually other constraints and/or objectives must be taken into account, in addition to the overlapping constraint and the volume maximization. For example, loading stability and weight distribution are important factors in some applications. Bischoff & Ratcliff (1995) list twelve factors that play an important role in container loading problems. Gehring *et al.* (1990) suggest a load arrangement that takes into account constraints in weight distribution. Gehring & Bortfeldt (1997) address the problem with several constraints, namely, orientation, load bearing, maximum weight, stability and weight distribution. Davies & Bischoff (1999) propose a heuristic that is able to produce loading arrangements which combine high space utilization with an even weight distribution of the cargo. Bortfeldt *et al.* (2003) and Mack *et al.* (2004) consider constraints of orientation and stability in terms of sections of the cargo that

overhang beyond the edge of the box(es) supporting it. He & Cha (2002) consider the sum of weighted objectives involving volume maximization, weight maximization and the minimization of the height of the gravity center. Bischoff (2006) develops an algorithm for tackling problems where the load bearing strength of the cargo is a key factor.

In this paper we put forward a multi-start random constructive heuristic for the container loading problem with a load arrangement that is based on maximal cuboids that can be loaded in given empty spaces, such that they fit in a single container and maximize its volume utilization subject to orientation and stability constraints.

At each step of the construction process one maximal cuboid is chosen probabilistically from a restricted list of candidates. In order to enhance the flexibility in the construction of a solution, we allow a probabilistic reduction on such cuboids. This approach can be viewed as a generalization of procedures that load one box at a time and the cuboid approach. The paper contains five sections. The following section defines the problem in more detailed terms. Section 3 of the paper describes the proposed multi-start heuristic, and in section 4 the heuristic performance is tested on standard benchmarks and additional instances suggested in the literature. Conclusions are presented in section 5.

2. Problem Description

The container loading problem involves the selection of a subset of boxes, each box with a given utility, such that they fit in a single container and maximize the total utility subject to a set of constraints. In this paper the utility of each box is its volume, and constraints of concern are orientation and stability. The boxes have $k = 1,..., 6$ orientations and are grouped in m types, each type t characterized by three spatial dimensions d_{1tk}, d_{2tk}, d_{3tk}, a volume v_t and a number q_t of boxes, $t = 1, ..., m$. The set of boxes is said to be homogeneous if it has a single type, while it is weakly heterogeneous or strongly heterogeneous if the number of types is small or large relative to the total number of items, respectively. Without loss of generality the dimensions of the container and of the boxes are positive integers. Constraints related to the orientation of the boxes are represented by a set of binary parameters u_{tk}, such that $u_{tk} = 1$ if the orientation a box of type t and an orientation k is allowed and $u_{tk} = 0$, otherwise. The stability constraint specifies that all loaded boxes are fully supported by the container floor or one or more the boxes.

In order to locate boxes in the container, consider a coordinate system within it such that when the container is viewed from the front, its origin is the bottom left corner of the container back, and the three coordinates refer to positions along the length, width and height, in that order. An empty space i in the container corresponds to a parallelepiped of dimensions $X_i \times Y_i \times Z_i$ and volume ve_i, whose coordinates (x_i, y_i, z_i) are associated to its lower left back vertex. When the container is empty, there exists a single empty space with the container dimensions and coordinates (0, 0, 0).

Figure 2 shows that for each box placed in the container, three new empty spaces are created: the depthwise, widthwise, and heightwise spaces. The intersection space of the depthwise and widthwise empty spaces is not considered as an empty space since it is contained in these spaces. An empty space is discarded if no unloaded box fits into the space.

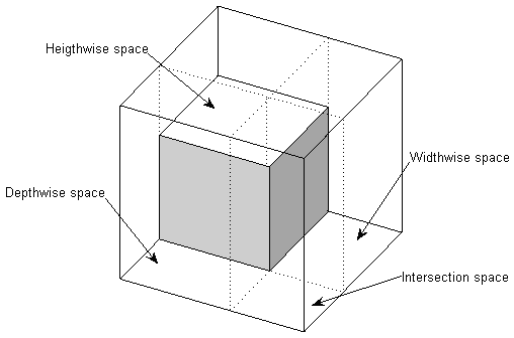
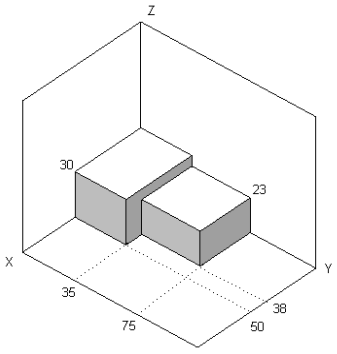


Figure 2 – Spaces generated by the loading of one box.

The proposed heuristic makes use of an adaptation of the spatial representation suggested by Ngoi *et al.* (1994), in which a two-dimensional matrix represents a view from the top of the container and contains cells that correspond to the height of potential loading surfaces. The search for available loading spaces amounts to scanning this matrix for contiguous surfaces at the same height. The adaptation is similar to that proposed by Bischoff (2006). The spatial representation technique provides excellent flexibility in generating the packing sequence and in identifying all potential placement locations, unlike the wall approach building used in several heuristics. In this paper, the identification of empty spaces in the container is performed in a single direction, from backward to forward, and in both directions, from backward to forward and from forward to backward.

Figure 3 illustrates an example of such a representation for a container with dimensions $90\text{ cm}\times120\text{ cm}\times100\text{ cm}$ and two boxes of dimensions $50\text{ cm}\times35\text{ cm}\times30\text{ cm}$ (box 1) and $38\text{ cm}\times40\text{ cm}\times23\text{ cm}$ (box 2), with coordinates $(0,0,0)$ and $(0,35,0)$, respectively. The first row and first column of the matrix contain the values of the projections of all vertical box/container edges onto the two horizontal axes, with the exception of cell (1, 1) that represents the height of the container. The figures of 35, 75 and 120 in row 1 correspond to the width of box 1, the sum of the widths of the two boxes, and the container width (projections of the edges onto the *y* axis). Analogously, the figures of 38, 50 and 90 represent the length of box 2, the length of box 1 and the container length (projections of the edges onto the *x* axis). The remaining cells correspond to the heights of the surfaces: cells (2, 2) and (3, 2) correspond to the height of box 1 and cell (2, 3) represents the height of box 2.



Matrix of surfaces

100	35	75	120
38	30	23	0
50	30	0	0
90	0	0	0

Figure 3 – Representation of loading surfaces.

3. A Multi-Start Randomized Constructive Heuristic

In this paper we propose a multi-start constructive heuristic for the container loading problem. Multi-start heuristics have been used in combinatorial optimization since the seminal paper by Lin & Kernighan (1973) for the symmetric traveling salesman and they represent a strategy to achieve search diversification in the solution space. One of the most well known multi-start methods is the greedy random adaptive search procedure (GRASP), which was introduced by Feo & Resende (1995). GRASP is a meta-heuristic that has been applied with success to solve a variety of combinatorial problems (Festa & Resende, 2004), and enhancement techniques to the basic GRASP are discussed in Resende & Ribeiro (2003). Each GRASP iteration consists of two phases: construction and local search. The construction phase builds a feasible solution by probabilistically selecting the next element to be incorporated in a partial solution from a restricted candidate list (RCL) composed of the best elements, as measured by a greedy function. Local search is then applied to the constructed solution until a local optimum is found. This process is repeated for a number of iterations, and the best local optimum is selected.

The proposed heuristic is based only on the construction phase due to the difficulty of devising a restricted neighborhood with promising moves that operate directly on the constructed solution. To the best of our knowledge, Faroe *et al.* (2003) is the only work that performs a local search directly on the loading solutions for a three-dimensional loading problem, which in this case is the three-dimensional bin packing problem. The neighborhood of a solution consists of all solutions that can be obtained by translating any single box along one of the coordinate axes or to the same position in another bin. In this way, solutions with overlapping of boxes are generated and hence the objective is to minimize the total volume of the pairwise overlap between boxes. An obvious drawback of this strategy is that the orientation of boxes of neighbor solutions does not change. The remaining papers that make use of local search for the container loading problem use a codification of the solution, and a neighbor is defined by a move applied to the coded solution (Bortfeldt & Gehring, 1998; Bortfeldt & Gehring, 2001; Bortfeldt *et al.*, 2003; Moura & Oliveira, 2005). Martí (2003) describes the best known multi-start heuristics and also remarks that for some problems it is more effective to construct solutions than apply a local search procedure.

The heuristic has some similar features with the two-step heuristic suggested by Eley (2002), which can be summarized as follows. In the first step, a greedy heuristic builds a solution by initially sorting the boxes by decreasing volume. All possible empty spaces for stowing the next box are examined, and this box is placed in the empty space where the sum of the volume of spare spaces that cannot filled by the remaining boxes is minimal. The second step of the heuristic consists of a tree search in which the root node represents an empty container and each further node that is not a leaf node constitutes a partially filled container. If all orientations are permitted for each box type at a given node, then each partial solution is branched into $6 \cdot m$ new partial solutions. A best search strategy is applied, which expands a number (a parameter) of nodes that obtained the highest ranking from the evaluation function, which is a lower bound derived by filling the remaining space of the corresponding partial solution by applying the greedy heuristic.

3.1 Heuristic Description

Cuboids are homogeneous blocks composed of boxes of the same type and orientation. A cuboid is represented by a triple $c_{tki} = (c^x_{tki}, c^y_{tki}, c^z_{tki})$, such that its elements denote the number of boxes of type t with orientation k in the coordinates x, y, z, that can be loaded in the empty space i. The number of boxes that form a cuboid is given by $|c_{tki}| = c^x_{tki} \cdot c^y_{tki} \cdot c^z_{tki}$. Figure 4 shows a cuboid $c_{tki} = (2, 2, 3)$ with $|c_{tki}| = 12$ boxes of type t and orientation k, with two boxes along its length and width and three boxes along its height. A single box is a cuboid with representation (1, 1, 1), and the volume of the cuboid is the product of the volume of the box type and the number of boxes that make up the cuboid, i.e., $v_t \cdot |c_{tki}|$.

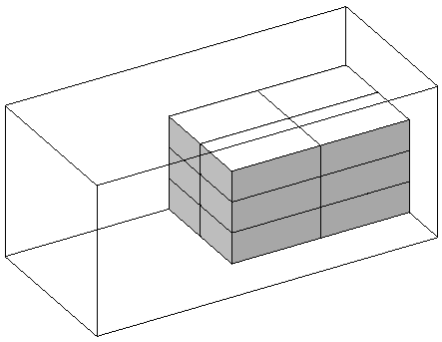


Figure 4 – A cuboid (2, 2, 3) composed of boxes of the same type and orientation.

Let $\overline{c}_{tki} = (\overline{c}^x_{tki}, \overline{c}^y_{tki}, \overline{c}^z_{tki})$ be the largest cuboid formed by boxes of type t and orientation k which can be assigned to an empty space i with dimensions $X_i \times Y_i \times Z_i$ and volume ve_i. The number of boxes which determines each dimension of the cuboid $\overline{c}_{tki}$ is calculated as follows:

$$\overline{c}^z_{tki} = \min\left(\left\lfloor \frac{Z_i}{d_{3tk}} \right\rfloor, \ \hat{q}_t \right) \tag{1}$$

$$\overline{c}^y_{tki} = \min\left(\left\lfloor \frac{Y_i}{d_{2tk}} \right\rfloor, \left\lfloor \frac{\hat{q}_t}{\overline{c}^z_{tki}} \right\rfloor\right) \tag{2}$$

$$\overline{c}^x_{tki} = \min\left(\left\lfloor \frac{X_i}{d_{1tk}} \right\rfloor, \left\lfloor \frac{\hat{q}_t}{\overline{c}^z_{tki}\,\overline{c}^y_{tki}} \right\rfloor\right), \tag{3}$$

where $\hat{q}_t$ denotes the number of boxes of type t that have not been loaded in the container.

Any cuboid c_{tki} such that $1 \le c^x_{tki} \le \overline{c}^x_{tki}$, $1 \le c^y_{tki} \le \overline{c}^y_{tki}$, and $1 \le c^z_{tki} \le \overline{c}^z_{tki}$ can compose the loading pattern. Therefore, the use of cuboids with variable size generates a constructive tree,

in which each node generates at most $\sum_{t=1}^{m}\sum_{k=1}^{6}|\overline{c}_{tki}|$ child nodes, which is a substantial increase relative to the maximum of $6 \cdot m$ new partial solutions in each node of the tree suggested by Eley (2002). In the example of Figure 4, the cuboid (2, 2, 3) generates 12 nodes. As a result, the search is conducted in a larger solution space, thus providing a more flexible approach to tackle weakly and strongly heterogeneous cargoes. At each node a bias function is used to prioritize the building of "good" cuboids.

The addition of the length with lateral support to the computation of the cuboid prevents a large fragmentation of the front space of the container and also improves the horizontal stability.

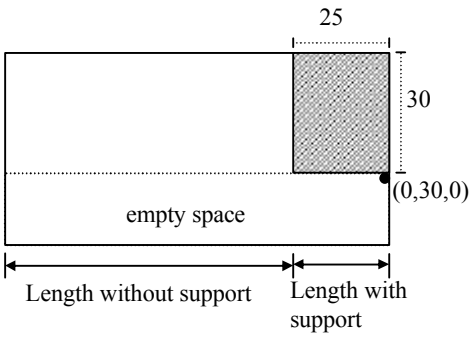


Figure 5 – Top view of empty spaces for a container with one box.

Figure 5 shows the top view of empty spaces with and without lateral support for a container with one box that has a base of $25\,\text{cm} \times 30\,\text{cm}$. The empty space with coordinates (0, 30, 0) has the same length as the container, from which the length of 25 cm has lateral support. Let s_i^x denote the length of an empty space i with lateral support. Then expression (3) is modified as follows:

$$c_{tki}^{x} = \min\left(\min\left(\left\lfloor \frac{X_i}{d_{1tk}} \right\rfloor, \left\lfloor \frac{s_i^x}{d_{1tk}} \right\rfloor \right), \left\lfloor \frac{\hat{q}_t}{c_{tki}^{z} c_{tki}^{y}} \right\rfloor \right) \tag{4}$$

Note that the second term in expression (4) yields the maximum number of boxes that can be loaded with lateral support.

Another strategy employed to reduce the fragmentation of the front space is to discard empty spaces that are considered *too far* from empty spaces that lie further back in the container (Verweij, 1996). Formally, let x_i and x_0 be the coordinates along the axis x of an empty space i and an empty space 0 further back in the container, respectively, and let $d_{1k_0t_0}$ be the largest length of the unloaded boxes. Then any empty space that satisfies $x_i - x_0 > d_{1k_0t_0}$ is considered *too far*, as illustrated in Figure 6.

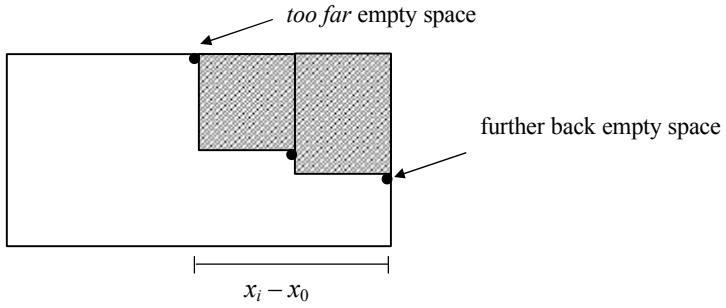


Figure 6 – Example of a *too far* empty space.

Similarly to the approach followed by Bischoff (2006), for each combination of box type, orientation and empty space, a candidate cuboid $\overline{c}_{tki}$ is evaluated by sets of the six criteria that are arranged in decreasing order of importance.

C1. Largest normalized space utilization of the empty space:

$$ad\left(X_i - \overline{c}^x_{tki}\, d_{1kt}\right) + ad\left(Y_i - \overline{c}^y_{tki}\, d_{2kt}\right) + ad\left(Z_i - \overline{c}^z_{tki}\, d_{3kt}\right)$$

where

$$ad\left(D - \overline{c}\, d\right) = \begin{cases} 1 & \text{if } D - \overline{c}\, d = 0 \\ 1 \big/ \left(D - \overline{c}\, d\right) & \text{otherwise,} \end{cases}$$

D represents one of the dimensions X_i, Y_i, Z_i of the empty space, and $\overline{c}\, d$ denotes the respective dimension of the packed boxes $\overline{c}^x_{tki}\, d_{1kt}, \overline{c}^y_{tki}\, d_{2kt}, \overline{c}^z_{tki}\, d_{3kt}$.

C2. Smallest difference between the height of the empty space and the height of the cuboid: $Z_i - d_{3tk}\overline{c}^z_{tki}$.

C3. Largest volume utilization of the empty space: $\dfrac{v_t \left|\overline{c}_{tki}\right|}{ve_i}$.

C4. Largest base area: $d_{1tk}\overline{c}^x_{tki} d_{2tk}\overline{c}^y_{tki}$.

C5. Smallest lenghtwise protrusion: $d_{itk}\overline{c}^x_{tki} + x_i$.

C6. Empty space with smallest widthwise coordinate: y_i.

Table 1 – Sets of evaluation criteria.

Set	Criteria
E_1	C1, C2, C5, C6
E_2	C3, C4, C5, C6
E_3	C1, C2, C4, C6
E_4	C3, C2, C4, C6

Table 1 shows four evaluation sets, each one containing four criteria, which are used to assess a candidate cuboid. The first criterion in each set is the most important and the remainders are tie-breakers. For each instance the most suitable evaluation set is determined probabilistically according to the mechanism proposed by Prais & Ribeiro (2000) for selecting the parameter that restricts the candidate list in the construction phase of GRASP. In the first K iterations of the heuristic, we collect information about the container volume utilization associated to the choice of each evaluation set $E_i, i = 1,2,3,4$, and the probabilities are made equal to p_i = 1/4. Let V^* denote the best current volume utilization and let $\overline{V}_i$ be the average volume of all solutions found by using the evaluation set $E_i, i = 1,2,3,4$. The selection probabilities are periodically revaluated every K iterations, by taking $p_i = l_i / \sum_{j=1}^{4} l_j$ with $l_i = (\overline{V}_i / V^*)^f$ for i = 1, 2, 3, 4. The parameter f may assume a value greater than one in order to accentuate the difference among l_i, i = 1, 2, 3, 4.

The t best cuboids, as evaluated by the above procedure are ranked from the best to the worst in a restricted candidate list (RLC). A cuboid in RLC is then selected with probability that is based on its rank $r(s)$ and a bias function that favors the selection toward some particular candidates, as suggested by Bresina (1996). Let $r(s)$ denote the rank of an element s in RLC and let bias($r(s)$) represent the bias function. The probability $p(s)$ of selecting element s is

$$p(s) = \frac{bias(r(s))}{\sum_{s' \in RLC} bias(r(s'))} \tag{5}$$

The number of boxes of the selected cuboid is then reduced probabilistically along the three dimensions by means of a bias function that favors values close to zero. Let qb be the number of boxes along one dimension and let $rd \in [0, qb-1]$ be the reduction number. Then the probability of selecting the reduction r along this dimension is given by

$$p(rd) = \frac{bias(rd)}{\sum_{rd'=0}^{qb-1} bias(rd')}. \tag{6}$$

Figure 7 shows a schematic representation of the steps that compose the proposed container loading heuristic. Initially, the empty spaces in the container are identified (a) and from the unloaded boxes (b), maximal cuboids are computed (c). After selecting a maximal cuboid, probabilistic reductions are applied to such a cuboid (d), and finally the cuboid is loaded (e). This procedure is repeated until no additional boxes fit in the container.

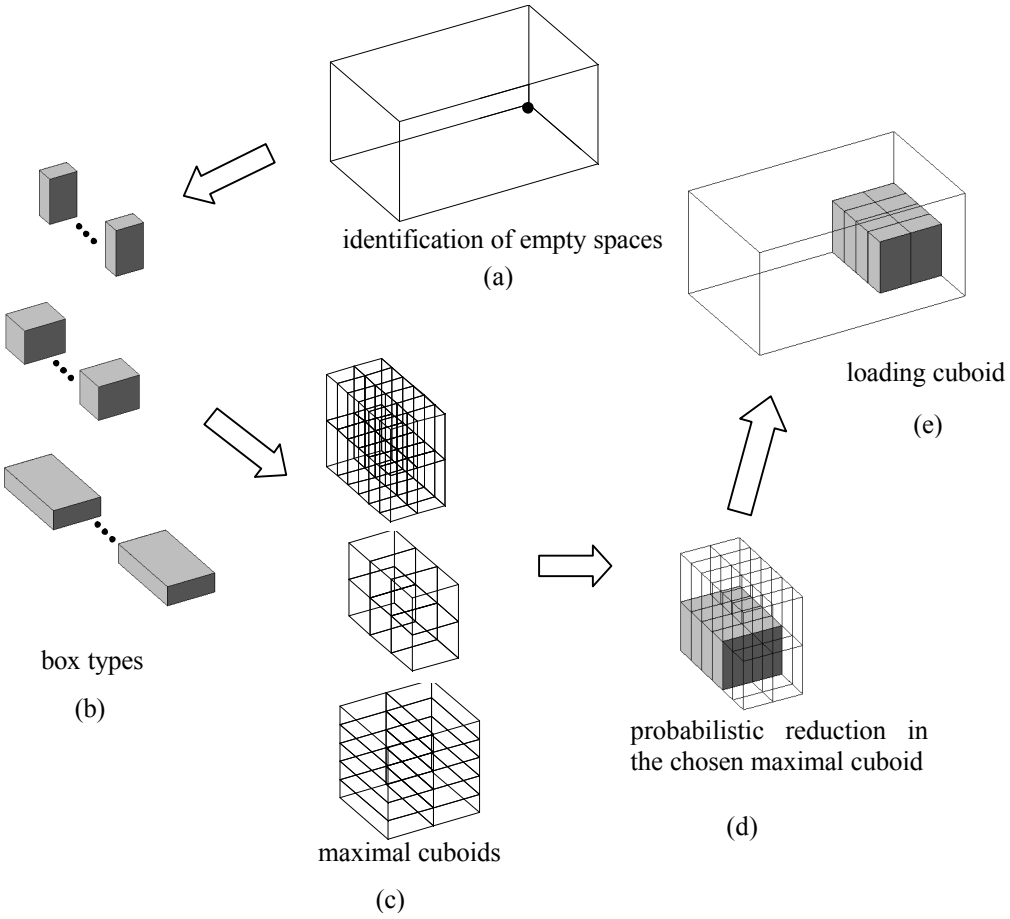


Figure 7 – Schematic representation of the heuristic.

Figure 8 presents the pseudo-code of the multi-start heuristic algorithm for the container loading problem. The variable V^* and the variable *iter*, which counts the number of iterations, are initialized in lines 1 and 2, respectively. The counter is updated in line 4. The procedure of choosing a set of evaluation criteria in line 5 is as presented above. Line 6 invokes the procedure that constructs a solution *S*. The best current solution (incumbent solution) is updated in line 7, and the incumbent solution is returned in line 9.

```
   Procedure multi-start algorithm for container loading
1  V* ← 0
2  iter ← 0
3  Repeat
4      iter ← iter + 1
5      Criteria ← choose a set of evaluation criteria
6      S ← Random_Construction (Criteria)
7      If V* < V^S then S_incumbent ¬ S End_if
8  Until stopping condition is met
9  Return(S_incumbent)
```

Figure 8 – Multi-start heuristic algorithm.

Figure 9 shows the pseudo-code of the random constructive heuristic. The structure that stores a solution S and the list of empty spaces are initialized in lines 1 and 2. The construction of a solution proceeds while there exist unloaded boxes and empty spaces, as indicated in line 3. Lines 4 to 16 correspond to the evaluation of the combination of empty spaces and box types with feasible orientations in order to build the RCL. Cuboids are calculated in lines 9, and in line 11, the elements of RCL are ordered according to the evaluation criteria. In line 17 an element of RCL is selected, in line 18 a reduction is applied to this element, which is then added to the partial solution in line 19. The lists of empty spaces and unloaded boxes are updated in line 20, while the complete solution in returned in line 22.

```
   Procedure Random_Construction (Criteria)
1  S ← {}
2  list_empty_spaces ← empty spaces of S
3  While Σ_{t=1}^{m} q̂_t > 0 AND |list_empty_spaces| > 0 do
4      For i ¬ 1 to |list_empty_spaces| do
5          For t ← 1 to m do
6              If q̂_t > 0 AND a box of type t fits in the empty space i then
7                  For k ← 1 to 6 do
8                      If the orientation k of Box of type t is feasible then
9                          Compute the cuboid c̄_tk according to expressions (1), (2) e (4)
10                     End_if
11                     Update RLC by using Criteria
12                 End_for
```

14	**End_if**
15	**End_for**
16	**End_for**
17	Select a cuboid from *RLC* according to the bias function (5)
18	Apply reductions to the selected cuboid by using the bias function (6)
19	Include the selected cuboid in the solution S
20	Update $list\_empty\_spaces$ and the number of unloaded boxes
21	**End_while**
22	**Return**(S)

Figure 9 – Algorithm for the random constructive heuristic.

4. Computational Experiments

The multi-start algorithm was coded in C++ and compiled with version 3.3.3 of the gcc compiler with the optimization flag –O3 and computational tests for sets of instances of the literature were carried out, unless otherwise stated, on a PC Intel Pentium IV 2.8 GHz with 512 Mb of RAM.

If the identification of empty spaces in the container is performed from backward to forward, the heuristic version is denoted AAR1 and in case it is carried out from backward to forward and forward to backward simultaneously, it is denoted AAR2. Unless otherwise stated, the maximum number of constructed solutions is 240.

From the bias functions suggested by Bresina (1996), we selected the same function in expressions (5) and (6) with the form $bias(x) = x^{-n}$ since the best candidate elements in RLC have close evaluation values and best reduction values should be close to zero. Let V^p denote the volume of a partial solution. The exponent n used in expressions (5) and (6) is then defined as

$$n = \begin{cases} n_1 & \text{if } V^p < r \cdot V^* \\ n_2 & \text{otherwise} \end{cases}$$

where, $r = 0{,}8$, $n_1 = 2$ e $n_2 = 3$ for instances with less than eight types of boxes and $n_1 = 3$ e $n_2 = 4$, otherwise. The remaining parameters were set to $\tau = 10$, $K = 100$ and $f = 10$.

The reported results for versions AAR1 e AAR2 correspond to the average throughout ten runs, with different seeds for the pseudo-random number generator. The performance of the multi-start heuristic is compared with that of seventeen heuristics:

- MA – heuristic AND/OR-graph (Morabito & Arenales, 1994)
- BJR – constructive heuristic (Bischoff *et al.*, 1995)
- BR – constructive heuristic (Bischoff & Ratcliff, 1995)
- BG_1 – genetic algorithm (Gehring & Bortfeldt, 1997)

- BG_2 – tabu search (Bortfeldt & Gehring, 1998)
- DB – constructive heuristic (Davies & Bichoff, 1999)
- BG_3 – hybrid genetic algorithm (Bortfeldt & Gehring, 2001)
- GB – parallel genetic algorithm (Gehring & Bortfeldt, 2002)
- E – constructive heuristic with tree search (Eley, 2002)
- BGM_1 – sequential tabu search (Bortfeldt *et al.*, 2003)
- BGM_2 – parallel tabu search (Bortfeldt *et al.*, 2003)
- CM – five constructive heuristics (Cecílio & Morabito, 2004)
- CD – constructive heuristic (Chien & Deng, 2004)
- MBG – hybrid tabu search/simulated annealing (Mack *et al.*, 2004)
- LZ – squeaky wheel optimization (Lim & Zhang, 2005)
- MO – GRASP (Moura & Oliveira, 2005)
- B – multi-start heuristic (Bischoff, 2006)

The code of Cecílio & Morabito (2004) was used in Tables 2, 3, 4 and 5 to obtain results for the heuristic CM. Table 2 shows the results for instances generated according to the scheme suggested by Cecílio & Morabito (2004), who also proposed five heuristics that correspond to refinements of the approach by George & Robinson (1980). The best result obtained by the application of such heuristics to each instance is reported. The numbers in parenthesis in the first column represent the number of box types. For all instances, the volume utilization provided by the version AAR2 outperforms AAR1, which in turn outperforms CM. However, AAR2 spends more computational time, followed by AAR1 and CM. The small computational time required by CM can be explained by the identification of a smaller number of empty spaces due to the use of vertical walls, and a greedy heuristic that selects one of the box types. Recall that algorithms AAR1 and AAR2 do not impose constraints on the identification of empty spaces and considers all possible combinations of empty spaces and box types for the selection of a cuboid that builds the load arrangement. If the number of solutions evaluated by AAR1 and AAR2 is reduced to 60, then the mean computational time is lowered to 0.13 seconds and 0.31 seconds, respectively. With this reduction, the mean volume utilization becomes 89.56% for AAR1 and 90.20% for AAR2, still maintaining the dominance ordering for all instances as mentioned above.

Table 2 – Results for instances of Cecílio & Morabito (2004).

Instances	CM		AAR1		AAR2	
	Vol. (%)	Time (s)	Vol. (%)	Time (s)	Vol. (%)	Time (s)
C1 (10)	89.25	0.16	93.81	0.21	**94.30**	0.45
C2 (10)	81.05	0.07	87.76	0.13	**88.07**	0.25
C3 (10)	90.78	0.18	94.31	0.14	**95.06**	0.36
C4 (10)	83.93	0.08	89.51	0.10	**89.90**	0.19
C5 (50)	86.56	0.43	91.00	2.02	**91.25**	4.66
C6 (50)	72.87	0.18	85.32	0.91	**85.49**	1.63
C7 (50)	87.19	0.36	92.39	1.04	**92.98**	2.69
C8 (50)	80.35	0.20	91.17	0.56	**91.54**	1.11
Mean	84.00	0.21	90.66	0.64	**91.07**	1.42

Table 3 presents the volume utilization percentage for twelve real instances relative to seven companies as described by Cecílio & Morabito (2004). The instances are grouped by company, for example, instances B-1 and B-2 refer to company B. For eight instances, all heuristics find optimal solutions, i.e., solutions in which all boxes are loaded. The heuristics AAR1 and AAR2 find an optimal solution for instance G-3, while the heuristic CM finds an optimal solution for instance G-1. The heuristic AAR1 obtains the best solutions for instances B-1 and G-2, for which optimal solutions are not available. Computational times to solve all instances are similar, being of order of hundredths of seconds for CM and tenths of seconds for AAR1 and AAR2.

Table 3 – Results for real instances of Cecílio & Morabito (2004).

Instances	CM Vol. (%)	AAR1 Vol. (%)	AAR2 Vol. (%)
A (47)	* **86.22**	* **86.22**	* **86.22**
B-1 (26)	85.56	**89.25**	88.67
B-2 (22)	* **83.67**	* **83.67**	* **83.67**
C (04)	* **79.99**	* **79.99**	* **79.99**
D (10)	* **75.02**	* **75.02**	* **75.02**
E (05)	* **80.73**	* **80.73**	* **80.73**
F-1 (05)	* **94.23**	* **94.23**	* **94.23**
F-2 (05)	* **92.59**	* **92.59**	* **92.59**
F-3 (05)	* **93.54**	* **93.54**	* **93.54**
G-1 (05) #	* **99.29**	96.51	97.90
G-2 (04)	98.38	**98.98**	98.96
G-3 (09)	89.02	* **89.80**	* **89.80**

* Optimal solution

We have not been able to reproduce this result with the code of Cecílio & Morabito (2004)

A PC Athlon 800 MHz, 512 RAM was used for the execution of computational tests of Tables 4 and 5. Table 4 shows the results obtained by the heuristics MA, CM, AAR1 and AAR2 for the 80 instances generated by Morabito & Arenales (1994). Such instances refer to the unconstrained container involving 5, 10, 20 and 30 types (in parentheses in the first column) of boxes with dimensions ranging from 5% to 85% of the container dimensions. The heuristic MA suggested by Morabito & Arenales (1994), which was adapted to allow rotations (Cecílio & Morabito, 2004), outperforms the remaining heuristics, though at a larger computational effort.

Since several boxes of the 80 instances have dimensions comparable to the container dimensions, it is reasonable to adjust the parameters of the bias functions (4) and (5) in order to obtain a higher degree of randomness, so that $n_1 = 1$ and $n_2 = 4$. The modified heuristic is denoted AAR1*. The stopping criterion now is computational time, which is similar to that reported by Morabito & Arenales (1994). Table 5 shows that the heuristics MA and AAR1 are very competitive and that the version AAR1* yields best solutions in six sets and a better mean volume utilization.

Table 4 – Results for instances of Morabito & Arenales (1994).

Instances	MA		CM		AAR1		AAR2	
	Vol. (%)	Time (s) ¥	Vol. (%)	Time (s)	Vol. (%)	Time (s)	Vol. (%)	Time (s)
s1 (05)	**88.02**	10.63	81.50	0.01	86.29	0.16	85.90	0.24
s2 (05)	**93.40**	12.81	83.86	0.01	90.78	0.13	90.70	0.18
s3 (05)	**83.54**	3.87	82.52	0.01	83.20	0.16	83.09	0.19
s4 (10)	**98.71**	40.78	94.73	0.03	97.29	0.16	97.29	0.20
s5 (05)	**95.21**	24.49	88.92	0.04	94.20	0.09	94.39	0.13
s6 (10)	**97.17**	66.34	94.51	0.01	96.38	0.16	96.46	0.19
s7 (20)	**98.12**	127.05	95.61	0.05	97.14	0.29	97.28	0.31
s8 (30)	**98.44**	147.33	96.70	0.02	98.16	0.36	98.16	0.43
Mean	94.08	54.16	89.79	0.02	92.93	0.19	92.91	0.23

¥ Coded in the Pascal language, Borland 7.0 on a Pentium III 800 MHz with 256 Mb of RAM

Table 5 – New results for instances of Morabito & Arenales (1994).

Instances	MA		AAR1		AAR1*	
	Vol. (%)	Time (s)	Vol. (%)	Time (s)	Vol. (%)	Time (s)
s1 (05)	88.02	10.63	88.24	10.00	**88.44**	10.00
s2 (05)	93.40	12.81	93.32	12.00	**93.59**	12.00
s3 (05)	83.54	3.87	86.05	4.00	**86.86**	4.00
s4 (10)	**98.71**	40.78	98.34	40.00	98.50	40.00
s5 (05)	95.21	24.49	**95.50**	24.00	95.48	24.00
s6 (10)	97.17	66.34	97.36	66.00	**97.41**	66.00
s7 (20)	98.12	127.05	98.34	125.00	**98.41**	125.00
s8 (30)	98.44	147.33	98.86	145.00	**98.89**	145.00
Mean	94.08	54.16	94.50	53.25	**94.70**	53.25

Table 6 shows comparative results with the heuristic proposed by Chien & Deng (2004) for eleven real instances of shipping companies in Taiwan. The heuristic version AAR2 obtains best results for all instances with a computational time inferior to 0.3 seconds, and in three of the ten runs it yields a solution for instance 10 that fills 100% of the container volume. The total container filling is also obtained in one of the ten runs of the heuristic version AAR1 for instances 9 and 10. Chien & Deng report computational times between 9.42 and 1,384.43 seconds.

Table 6 – Results for instances of Chien & Deng (2004).

Heuristic	Instances										
	1	2	3	4	5	6	7	8	9	10	11
CD ¶	83.08	95.22	83.43	87.68	80.49	81.16	91.30	90.55	95.55	93.08	92.02
AAR1	83.94	97.26	93.29	98.66	**83.04**	**85.25**	99.56	**92.40**	98.09	99.57	97.57
AAR2	**83.94**	**97.32**	**94.73**	**98.64**	**83.04**	**85.25**	**99.74**	**92.40**	**99.16**	**99.82**	**97.81**

¶ Coded in the Matlab programming language version 6.1 on a Pentium II 350 MHz with 256 Mb of RAM

Table 7 lists heuristics and metaheuristics from the literature that utilize 700 instances generated by Bischoff & Ratcliff (1995) in order to validate their performance. These instances are divided into seven test cases BR1-BR7, each case with number of distinct boxes types (in parenthesis). Orientation and stability constraints are imposed and the box stability limit is set to 2. This implies that for a box of type t with orientation k, $u_{tk} = 0$ if $d_{3tk} / \left(\min_{j=1,2} d_{jtk} \right) \geq 2$, i.e., orientation k is not allowed if its height is greater than the double of one of the base dimensions. The results in this table are presented in increasing order of the mean volume occupation over the seven instances. The heuristic version AAR2 obtains best results for all test cases followed by the version AAR1. The mean computational times spent by AAR2 and AAR1 are 0.29 seconds and 0.14 seconds, respectively. The distinct computers used in the execution of the heuristics of Table 7 make it difficult to compare the computational time required by them in order to solve the instances. Nevertheless, it is worth stressing that the mean time spent by the metaheuristic MO on a Pentium IV 2.4 GHz with 480 MB of RAM is 33.5 seconds.

Table 7 – Results for instances of Bischoff & Ratcliff (1995).

Heuristic	Instances							Mean
	BR1(03)	BR2(05)	BR3(08)	BR4(10)	BR5(12)	BR6(15)	BR7(20)	
BJR	81.76	81.70	82.98	82.60	8276	81.50	80.51	81.97
BR	83.79	84.44	83.94	83.71	83.80	82.44	82.01	83.45
DB	84.10	84.50	85.00	84.70	84.60	83.70	82.70	84.19
BG_1	85.80	87.26	88.10	88.04	87.86	87.85	87.68	87.51
CM	89.05	87.40	87.21	86.75	87.09	86.05	84.82	88.34
E	88.05	88.44	89.23	89.24	88.99	88.91	88.36	88.75
MO	89.07	90.43	90.86	90.42	89.57	89.71	88.05	89.07
LZ	87.4	88.7	89.3	89.7	89.7	89.7	89.4	89.13
AAR1	90.86	90.88	90.94	90.67	90.40	90.14	89.46	90.48
AAR2	**91.73**	**91.60**	**91.47**	**91.06**	**90.90**	**90.46**	**89.54**	**90.96**

Bischoff & Ratcliff (1995) propose a heuristic that is designed to produce patterns which combine high space utilization with a high degree of stability, and suggest two measures related the stability. Measure 1 is the average number of boxes that support boxes that do not lie on the floor (the higher the better), while measure 2 is the average percentage of boxes not surrounded on at least three sides (the lower the better). Table 8 presents comparative results with six heuristics and metaheuristics that reported results for such measures. With regards to measure 1, BJR outperforms the other methods, while AAR1 is superior relative to measure 2, due to the cuboid approach. As expected, the performance of AAR2 is worse than AAR1, since in the first version the loading is carried out from backward to forward and forward to backward simultaneously, which results in empty spaces at the junction of the backward and forward fronts. However, this difficulty can be minimized by using a procedure of compacting the boxes.

Table 8 – Load stability for instances of Bischoff & Ratcliff (1995).

Heuristic	Instances													
	BR1		BR2		BR3		BR4		BR5		BR6		BR7	
	M1	M2	M1	M2	M1	M2	M1	M2	M1	M2	M1	M2	M1	M2
BJR	2.02	8.50	2.22	11.21	2.20	15.93	2.10	17.51	2.09	21.60	2.04	22.13	1.92	27.07
BR	1.13	10.36	1.10	14.60	1.08	19.67	1.07	23.53	1.06	26.03	1.06	31.04	1.04	35.99
BG_1	-	11.00	-	16.00	-	18.50	-	21.50	-	22.50	-	25.00	-	28.50
CM	1.14	7.57	1.12	10.75	1.10	13.72	1.10	14.99	1.10	16.50	1.10	19.58	1.10	21.76
E	-	9.80	-	13.50	-	18.00	-	20.50	-	21.50	-	22.90	-	26.00
MO	1.07	11.53	1.10	12.67	1.09	17.75	1.10	20.03	1.10	22.75	1.10	26.50	1.11	28.86
AAR1	1.15	6.00	1.15	9.22	1.12	10.35	1.11	12.48	1.11	13.70	1.10	15.70	1.08	18.24
AAR2	1.18	10.77	1.18	13.72	1.15	16.17	1.13	18.09	1.13	19.51	1.12	20.91	1.10	23.91

Table 9 shows results in increasing order of the mean volume occupation obtained by additional metaheuristics from the literature for the 700 instances BR1-BR7. The authors of the metaheuristics BGM_1, BGM_2 and MBG report only the mean volume occupation for fully supported boxes. For this experiment we stipulated computational time as the stopping criterion for AAR1 and AAR2. The heuristic version AAR2 obtains best results for all test cases followed by the version AAR1, with the exception of test case BR7. Note that the proposed approach is more effective for weakly heterogeneous box sets. The time to solve all instances of each group is shown in the last line of the table. The mean computational time to solve the 700 instances is 52 seconds, and based on data available at http://www.spec.org/cpu2000/results/cfp2000.html we concluded that this time corresponds to approximately 75 seconds in an Intel Pentium 2 GHz. This computer was used by Mack *et al.* (2004) who report a mean computational time of 205 seconds to solve the 700 instances BR1-BR7.

Table 9 – Additional results for instances of Bischoff & Ratcliff (1995).

Heuristic	Instances							Mean
	BR1(03)	BR2(05)	BR3(08)	BR4(10)	BR5(12)	BR6(15)	BR7(20)	
BG_3	87.81	89.40	90.48	90.63	90.73	90.72	90.65	90.06
GB	88.10	89.56	90.77	91.03	91.23	91.28	91.04	90.43
BG_2	92.41	92.33	91.97	91.26	90.40	89.57	88.18	90.87
B	90.57	90.84	91.43	91.21	91.25	91.04	90.81	91.02
BGM_1	-	-	-	-	-	-	-	91.60
BGM_2	-	-	-	-	-	-	-	92.20
MBG	-	-	-	-	-	-	-	92.41
AAR1	92.58	92.93	92.96	92.70	92.48	92.11	**91.53**	92.53
AAR2	**93.38**	**93.25**	**93.11**	**92.77**	**92.51**	**92.16**	91.34	**92.64**
Time (s)	6.00	12.00	36.00	60.00	70.00	80.00	100.00	52.00

We have also tested the performance of the proposed heuristic on a real life example reported by George & Robinson (1980), which consists of 784 boxes distributed in eight types to be loaded in a container with available dimensions in millimeters $5793 \times 2236 \times 2261$. For the

stopping criterion of 240 evaluated solutions, AAR1 and AAR2 produce solutions in which all boxes are packed. We have extended this example by increasing one additional box of each type. In this case, AAR1 and AAR2 pack all boxes in 5 and 10 executions, respectively. When we consider two additional boxes of each type, AAR1 and AAR2 pack all boxes in 3 executions, respectively. Finally, AAR2 packs 24 additional boxes (three additional boxes of each type) in one execution with a container volume utilization of 93.03%.

5. Conclusions

In this paper we proposed a multi-start random constructive heuristic for loading boxes in a single container, with the objective of maximizing its volume utilization subject to orientation and stability constraints. Several conclusions can be drawn from the design and experiments for the proposed heuristic. Initially, the cuboid arrangement is an effective approach, even when dealing with a rather strongly heterogeneous set of boxes. In addition, the use of the adaptation of the spatial representation is very important to identify empty spaces. Finally, we have learned that a constructive heuristic with a controlled degree of randomization coupled with a suitable bias function is competitive and simpler when compared with metaheuristics proposed in the literature for this problem. The proposed heuristic is fairly robust, has few parameters and it is able to produce high quality solutions in short computational time. Further research involves the use of the proposed approach to deal with other practical constraints, such weight distribution and limited load bearing strength.

Acknowledgements

This research was partially funded by the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) and the Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP). The authors are also grateful to Fabiana Oliveira Cecilio and Reinaldo Morabito for providing us with the code of their algorithms.

References

(1) Bischoff, E.E. & Marriott, M.D. (1990). A Comparative Evaluation of Heuristics for Container Loading. *European Journal of Operational Research*, **44**, 267-276.

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

(3) Bischoff, E.E. (2006). Three-dimensional packing of items with limited load bearing strength. *European Journal of Operational Research*, **168**(3), 952-966.

(4) Bischoff, E.E.; Janetz, F. & Ratcliff, M.S.W. (1995). Loading Pallets with Non-Identical Items. *European Journal of Operational Research*, **84**, 681-692.

(5) Bortfeldt, A. & Gehring, H. (1998). Ein Tabu Search-Verfahren für Containerbelade-probleme mit schwach heterogenem Kistenvorrat. *OR Spektrum*, **20**(4), 237-250.

(6) Bortfeldt, A. & Gehring, H. (2001). A Hybrid Genetic Algorithm for the Container Loading Problem. *European Journal of Operational Research*, **131**, 143-161.

(7) Bortfeldt, A.; Gehring, H. & Mack D. (2003). A parallel tabu search algorithm for solving the container loading problem. *Parallel Computing*, **29**(5), 641-662.

(8) Bresina, J. (1996). Heuristic-Biased Stochastic Sampling. **In**: *Proceedings* of the 13th National Conference on Artificial Intelligence, 271-278.

(9) Cecilio, F.O. & Morabito, R. (2004). Refinamentos na heurística de George e Robinson para o problema de carregamento de caixas dentro de contêineres. *Transportes*, **12**(1), 32-45.

(10) Chen, C.S.; Lee, S.M. & Shen, Q.S. (1995). An Analytical Model for the Container Loading Problem. *European Journal of Operational Research*, **80**, 68-76.

(11) Chien, C.F. & Deng, J.F. (2004). A container packing support system for determining and visualizing container packing patterns. *Decision Support Systems*, **37**(1), 23-34.

(12) Davies, A.P. & Bischoff, E.E. (1999). Weight Distribution Considerations in Container Loading. *European Journal of Operational Research*, **114**, 509-527.

(13) Dyckhoff, H. (1990). A typology of cutting and packing problems. *European Journal of Operational Research*, **44**, 145-159.

(14) Eley, M. (2002). Solving container loading problems by block arrangement. *European Journal of Operational Research*, **141**(2), 393-409.

(15) Faroe, O.; Pisinger, D. & Zachariasen, M. (2003). Guided Local Search for the Three-Dimensional Bin-Packing Problem. *INFORMS Journal on Computing*, **15**(3), 267-283.

(16) Feo, T.A. & Resende, M.G.C. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, **6**, 109-133.

(17) Festa, P. & Resende, M.G.C. (2004). An annotated bibliography of GRASP. *European Journal of Operational Research*, submitted.

(18) Gehring, H. & Bortfeldt, A. (1997). A Genetic Algorithm for Solving the Container Loading Problem. *International Transactions in Operations Research*, **4**(5-6), 401-418.

(19) Gehring, H.; Menschner, K. & Meyer, M. (1990). A Computer-Based Heuristic for Packing Pooled Shipment Containers. *European Journal of Operational Research*, **44**, 277-288.

(20) George, J.A. & Robinson, D.F. (1980). A Heuristic for Packing Boxes into a Container. *Computers and Operations Research*, **7**, 147-156.

(21) Haessler, R.W. & Talbot, F.B. (1990). Load Planning for Shipments of Low Density Products. *European Journal of Operational Research*, **44**, 289-299.

(22) Hassamontr, J. (2003). On decomposing 3D Packing Problem in Wooden Furniture Industry. **In**: Proceedings 2003 IEEE International Symposium on Computational Intelligence in Robotics and Automation, 497-502.

(23) He, D.Y. & Cha, J.Z. (2002). Research on Solution to Complex Container Loading Problem Based on Genetic Algorithm. **In**: *Proceeding of First International Conference on Machine Learning and Cybernetics*, **1**, 78-82, Beijing, November.

(24) Hifi, M. (2004). Exact algorithms for unconstrained three-dimensional cutting problems: a comparative study. *Computers & Operations Research*, **31**(5), 657-674.

(25) Lai, K.K.; Xue, J. & Xu, B. (1998).Container packing in a multi-customer delivering operation. *Computers & Industrial Engineering*, **35**(1-2), 323-326.

(26) Lim, A. & Zhang, X. (2005). The container loading problem. *ACM Symposium on Applied Computing*, 913-917.

(27) Lim, A.; Rodrigues, B. & Wang, Y. (2003). A multi-faced buildup algorithm for three-dimensional packing problems. *Omega*, **31**(6), 471-481.

(28) Lin, S. & Kernighan, B.W. (1973). An Effective Heuristic Algorithm for the Travelling-Salesman Problem. *Operations Research*, **21**, 0498-0516.

(29) Mack, D.; Bortfeldt, A. & Gehring, H. (2004). A parallel hybrid local search algorithm for the container loading problem. *International Transaction in Operational Research*, **11**, 511-534.

(30) Martello, S.; Pisinger, D. & Vigo, D. (2000). The Three Dimensional Bin Packing Problem, *Operations Research*, **48**, 256-267.

(31) Martí, R. (2003). Multi-Start Methods. **In**: *Handbook of MetaHeuristics* [edited by F. Glover and G. Kochenberger], Kluwer, 355-368.

(32) Morabito, R. & Arenales, M. (1994). An And/Or-graph Approach to the Container Loading Problem. *International Transactions in Operations Research*, **1**(1), 59-73.

(33) Moura, A. & Oliveira, J.F. (2005). A GRASP Approach to the Container-Loading Problem. *IEEE Intelligent Systems*, **4**(20), 50-57.

(34) Ngoi, B.K.A.; Tay, M.L. & Chua, E.S. (1994). Applying spatial representation techniques to the container packing problem. *International Journal of Production Research*, **32**, 111-123.

(35) Pisinger, D. (2002). Heuristics for the container loading problem. *European Journal of Operational Research*, **141**, 143-153.

(36) Prais, M. & Ribeiro, C.C. (2000). Reactive GRASP: An application to a matrix decomposition problem in TDMA traffic assignment. *INFORMS Journal on Computing*, **12**, 164-176.

(37) Resende, M.G.C. & Ribeiro, C.C. (2003). Greedy randomized adaptive search procedures. **In**: *Handbook of MetaHeuristics* [edited by F. Glover and G. Kochenberger], Kluwer, 219-249.

(38) Verweij, A.M. (1996). Multiple destination bin packing. UU-CS (Ext. r. no. 1996-39). Utrecht, the Netherlands: Utrecht University: Information and Computing Sciences.