APPLYING HEURISTICS ON INTEGER LINEAR PROGRAMMING FOR SOLVING THE CONTAINER LOADING PROBLEM

Luiz Jonatã Pires de Araújo
University of Fortaleza (UNIFOR)
Av. Washington Soares, 1321 - J30, 60.811-905
Fortaleza - Brazil
ljonata@gmail.com

Plácido Rogério Pinheiro
University of Fortaleza (UNIFOR)
Av. Washington Soares, 1321 - J30, 60.811-905
Fortaleza - Brazil
placido@unifor.br

ABSTRACT

The Container Loading Problem consists, in this approach, in how to maximize the loaded cargo in a single container. We apply packing heuristics on integer linear programming to construct walls, layers and blocks of boxes. A backtracking algorithm chooses the best heuristics that make up the solution. We have used a well-known benchmark test suite to compare our results with other approaches. This paper also presents a case study of our implementation using some real data from ESMALTEC, a stove and refrigerator manufacturer in Brazil. The results of our approach represented a significant improvement over the current practice within ESMALTEC.

KEYWORDS. Container Loading Problem. Heuristics. Integer Linear Programming. PO na Indústria (IND).

1. Introduction

In recent years a significant amount of research has been devoted to variants of the Container Loading Problem (CLP). For example the Knapsack Container Loading Problem consists of packing boxes of various sizes within one or more containers, while optimizing a criterion such as the total loaded volume in a single container or the quantity of containers used. Some recent papers have also dealt with a weight distribution constraint.

This work describes a novel backtracking heuristic for the CLP, which attempts to pack the maximum volume of boxes with dimensions (l_i, w_i, h_i), and quantity b_i, $i = 1, \ldots, m$, into a single container with dimensions (L, W, D). Additional constraints include space limitations (no overlapping) and a weight limit. Thus, we are concerned with the maximization of the packed volume.

Instances of the CLP are typically classified according to the diversity of box types that can be loaded into the container. According to Dyckhoff (1990) and Pisinger (2002) the cargo can be regarded weakly heterogeneous (many items of relatively few different box types) or strongly heterogeneous (many items of many different box types). We evaluate our proposed algorithm on both homogeneous and weakly heterogeneous instances.

The paper is organized as follows. We present in Section 2 the related work. In Section 3, we show our methodology. In Section 4, we list the computational results. Finally, we draw conclusions regarding the quality of the solutions provided by our algorithm, as compared to those provided by other approaches, and make some considerations concerning future development.

2. Related Works

The CLP is a type of cutting and packing problems cf. the typology introduced by Dyckhoff (1990) and Wäscher *et al* (2007). The CLP is admittedly a NP-hard problem.

However, an instance of the CLP can be described as an integer programming model, such that of Chen, Lee, and Shen (1993). That model has $O(n^2)$ number of constraints or variables and applying a solver to directly solve it is impractical, for any reasonably sized instance. Chen, Lee, and Shen (1993) used that model to solve problems with up to 6 boxes. Martello, Pisinger, and Vigo (2000) presented an exact branch-and-bound method for solving CLP instances with up to 90 boxes. Although it finds the optimal solution, the implementation only works for small problems and requires high execution running time.

In face of the impracticality of the exact methods available for solving the CLP, many works proposed strategies, heuristics, for avoiding direct solution and improving the execution time. One of several heuristics proposed in the literature involves the building of box blocks, called layers (which can be vertical or horizontal), in which the dimensions of a layer are determined by the first box in the block, c.f. Bortfeldt and Gehring (2001) and George and Robinson (1980). Another relevant work is Pisinger (2002) that uses a tree search to decide the size of the layers. Morabito and Arenales (1994) present a guillotine cuts to split the empty space in the container and to fill it.

In addition to the proposed heuristics, other authors use metaheuristics such as Tabu Search, for example Bortfeldt, Gehring, and Mack (2002), or Genetic Algorithm (GA). For example Gehring and Bortfeldt (1997), that built stacks to the top of the container. Then, applying a GA to determine the order of placement of stacks and choose the configuration which results in the best use of container.

We increasingly find proposals that seek to combine the best of the exact method with heuristic methods. These ideas fall under a category of algorithm that has been commonly referred to as hybrid methods. A successful work along these lines was Nepomuceno, Pinheiro

and Coelho (2007), that generate reduced instances of the problem using genetic algorithms and solve these subproblems by linear programming. That proposal achieved an average allocation between 92 and 95%, but in very large running time. Another work was presented by Mack, Bortfeldt, and Gehring (2004), which combined Parallel Tabu Search with Simulated Annealing.

There is not a single approach that is the best for all problem types. Each one of the heuristics is better or worse in specific cases or instances of problems.

3. Methodology

This work is inspired on Pisinger (2000), which makes an allocation of layers coordinated by means by a tree search in order to determine the best depth or width of a layer. The backtracking algorithm navigates through the tree search to determine the best size for each layer. In other hand, our algorithm tries to determine the best heuristics instead of the best dimensions.

Data Structure The algorithm builds a tree that consists of nodes for controlling the processing flow, as presented in Fig. 1. The root node receives the input problem of the algorithm, that is to say, a definition of the empty space inside the container and a list of available boxes for packaging. Each node tests a heuristic to do some packing. If it cannot apply the heuristic, it attempts to apply another heuristic. The result of the successful application of a heuristic is a list of packed boxes and an output problem, i.e., the definition of a residual space to be filled and a new list of boxes. This output problem is the input problem to a new node in the tree, hierarchically below the node that preceded it. This process continues until we can no longer pack any box, regardless of the heuristic.

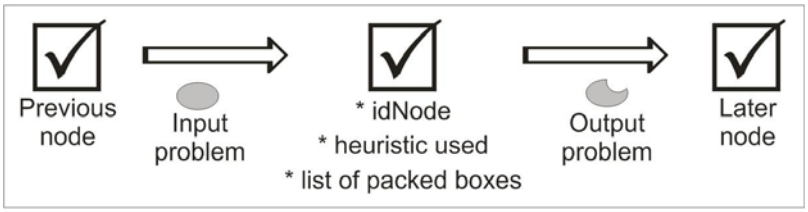

Fig. 1 A node and its role in our algorithm.

The algorithm maintains a tree, a set of nodes and their attributes. This tree is a solution of the input problem. Fig. 2 depicts two examples of resolutions found for a certain input problem.

Another data structure in our implementation is the box type, set of boxes with the same characteristics. Each box type is associated with a coefficient, which we call *relevance*, utilized for sorting the box types according to their volume. The higher the volume, the greater the relevance. This constant will allow us to prioritize the large packing boxes, leaving the smaller boxes (easier to be coefficient) to the end of the process, when residual space is small.

The function used in our implementation to calculate the relevance coefficient of the i^{th} greatest box type in a list of n box types:

$$r(i,n) = 2^{n-i}$$

In next topic, we present the using of the relevance coefficient.

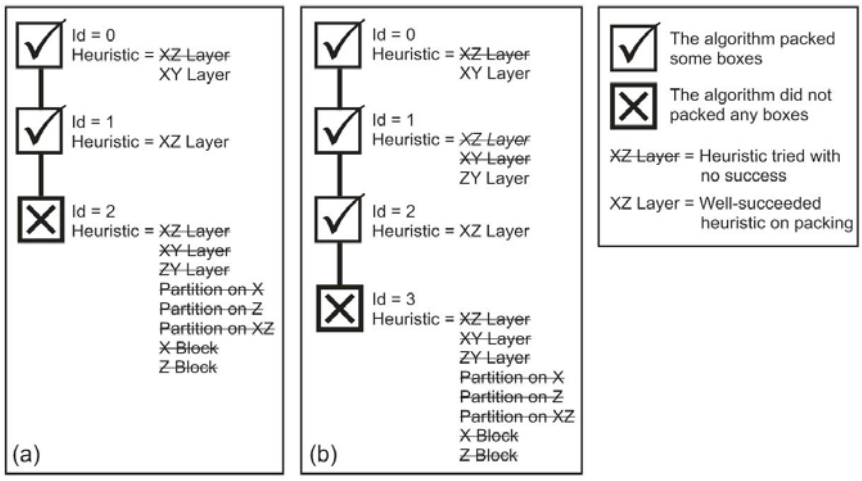

Fig. 2 (a) First found solution. (b) Better solution for the same problem.

Heuristics We discussed in Section 2 some popular heuristics (e.g. construction of layers, blocks or stacks) that are better or worse for certain problem types. In our algorithm, each heuristic resolves a mathematical model in integer linear programming. Each of these models has O(n) variables and about the same amount of constraints. Therefore, the application of a heuristic on an instance of the problem shows up extremely fast.

We illustrate how one of the heuristics used, the XZ Layer as presented in Fig. 3-a, works. Using this heuristic we intend to maximize the total relevance of packed boxes, not necessarily the volume. First, we select the type of boxes to be used in the model. Criterions:

- Little waste in Z-axis: The algorithm verify if, for the box type s, $(D\%d_s)/D \leq 0{,}04$. D is the depth of the container and d_s is the depth of the box type s. The value of 0,04 is the *tolerance parameter* for little waste used in our implementation. The *tolerance parameter* for little waste can be as close to zero as we intend.
- The height of layer is multiple of the selected box type: The algorithm verify if, for the box type s, $h_l \% h_s = 0$, that is, the remainder of division of the height of the layer (presented as h_l, the input parameter to this heuristic) by the height of the box type s (presented as h_s) should be equals to zero. In Fig. 3-a, for the first box type, $h_l / h_1 = 3$ and $h_l \% h_1 = 0$.

The input parameter h_l, the height of the XZ Layer, is the minimum common multiple of a subset of box types' heights. There are at most $2^m - 1$ possible values to this parameter, where *m* is the number of different heights of boxes types.

For each select box type *s* we calculate two constants:

- $nz_s = D / d_s$, nz_s integer. D is the depth of the container and d_s is the depth of the box type s;
- $ny_s = h_l / h_s$, ny_s integer. h_l is the height of the layer (input parameter to this heuristic) and h_s is the height of the box type *s*.

As follow the integer linear programming model:

$$\text{Max} \sum_{s \in S} (nx_s * nz_s * ny_s * r_s)$$

Subject to:

$$(\sum_{s \in S} nx_s * nz_s * ny_s) \leq q_s \qquad \forall s \qquad \text{R1}$$

$$\sum_{s \in S} nx_s \geq 1 \qquad \text{R2}$$

$$\sum_{s \in S} nx_s * w_s \geq W * 0{,}96 \qquad \text{R3.1}$$

$$\sum_{s \in S} nx_s * w_s \leq W \qquad \text{R3.2}$$

Where S is the set of all select box types and W is the weight of the container. r_s is the relevance, q_s is the quantity and w_s is the weight of the box type s. nz_s is the number of boxes (from the box type s) along the Z-axis. ny_s is the number of boxes (from the box type s) along the Y-axis. nx_s is the variable in this integer linear programming model and represents the number of boxes (from the box type s) along the X-axis.

The objective function intends to maximize the total relevance on the packed boxes.

The R1 constraint guarantees that there are enough boxes to build the layer. The R2 constraint guarantees at least on box type should be used.

The R3.1 and R3.2 constraints guarantee the little waste in X-axis, between up to 4%.

If there is not viable solution to the presented model, the algorithm considers that this heuristic was not succeeding for the input problem. So the algorithm tries another heuristic.

All heuristics used in the algorithm are: Layer XZ (a), Layer XY (b), Layer ZY (c), Partition on X (d), Partition on Z (e), Partition on XZ - Stack (f), Block on X (g) and Block on Z (h). They are all illustrated in Fig. 3.

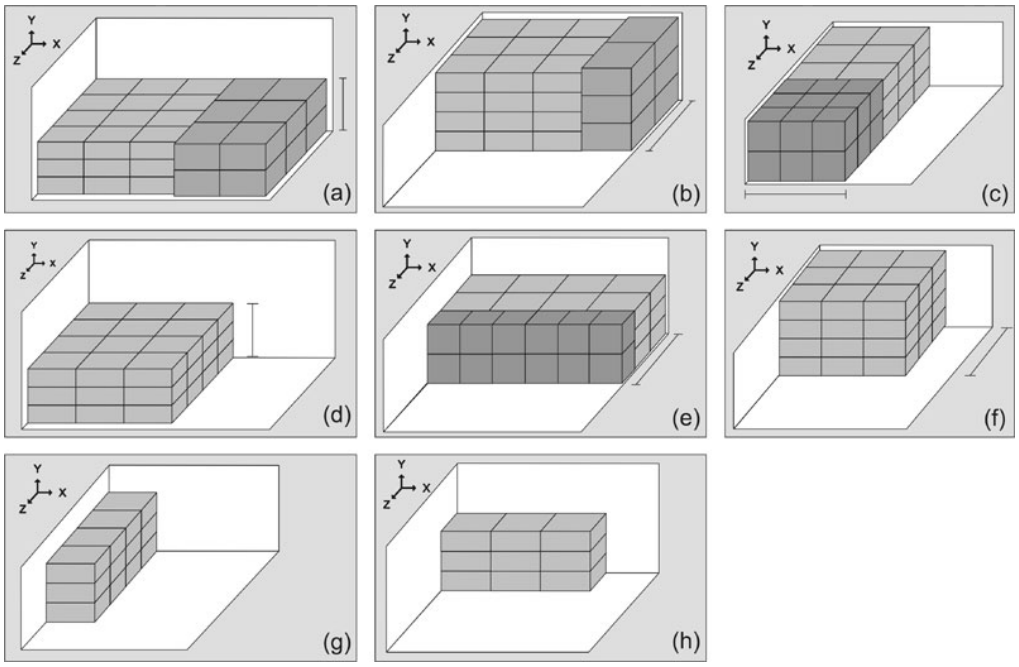

Fig. 3 The heuristics used in the algorithm.

We list in Table 1 some general characteristics of each heuristic, the objective function and needed parameters. Each one resolves an Integer Linear Programming Model.

Table 1 Characteristics of the used heuristics.

Heuristic	Objective function	Characteristics	Parameter
XZ Layer	Max. loaded **relevance**	- **Horizontal** layer; - Little waste of space in **X-axis** and **Z-axis**; - Stacks, even different types of boxes, must have the same **height**;	Layer's height
XY Layer	Max. loaded **relevance**	- **Vertical** layer; - Little waste of space in **X-axis** and **Y-axis**; - Ranks, even different types of boxes, must have the same **width**.	Layer's width
ZY Layer	Max. loaded **relevance**	- **Vertical** layer; - Little waste of space in **Z-axis** and **Y-axis**; - Ranks, even different types of boxes, must have the same **depth**.	Layer's depth
Partition on X	Max. loaded **relevance**	- **Horizontal** block; - Little waste of space in **Z-axis**; - Stacks, even different types of boxes, must have the same **height**.	Layer's height
Partition on Z	Max. loaded **relevance**	- **Horizontal** block; - Little waste of space in **X-axis**; - Stacks, even different types of boxes, must have the same **height**.	Layer's height
Partition on XZ (Stack)	Max. loaded **relevance**	- **Stack** block; - Little waste of space in **Y-axis**; - Stacks, even different types of boxes, must have the same **height** and **depth**.	Layer's width and depth
Block on X	Max. loaded **volume**	- **Horizontal** block with just one column of boxes; - Just one type of box.	None
Block on Z	Max. loaded **volume**	- **Horizontal** block with just one line of boxes; - Just one type of box.	None

The expression "little waste in X-axis", for example, means the container is almost full in dimension X, between 96% and 100% (values used in our implementation).

Backtracking In Fig. 2-a, we could notice that in the second node the algorithm was successful using the first heuristic (XZ Layer). But the heuristic that would lead to a better use of the container at a later node, in that specific case, was the third heuristic, as illustrated in Fig. 2-b. Therefore, a bad solution may be due to a bad choice of a heuristic in the previous node. We use backtracking heuristics to find the best solution though tree states.

4. Computational Results

We use a benchmark test suite to compare the quality of the solutions found by our algorithm to those found by other approaches. We also present a case study in which we compared our results with those in use within ESMALTEC, a stove and refrigerator manufacturer in Brazil.

The computational results were obtained using an Intel Core 2 Duo 2.1 GHz with 3 GB of RAM. The operating system is Windows Vista Home Edition. The development platform is Java 6.0 and Eclipse 3.1.1 tool. The used solver was CPLEX 9.0.

Benchmarking problems We have used the collection of test data sets for a variety of Operational Research problems, originally described by Beasley (1990). These data sets contain several types of problems, from homogeneous to strongly heterogeneous. The library is divided into files named BR1 to BR5. In BR1 library, the problems have 3 types of boxes (weakly heterogeneous problems). In BR2, there are 5 types of boxes and so forth.

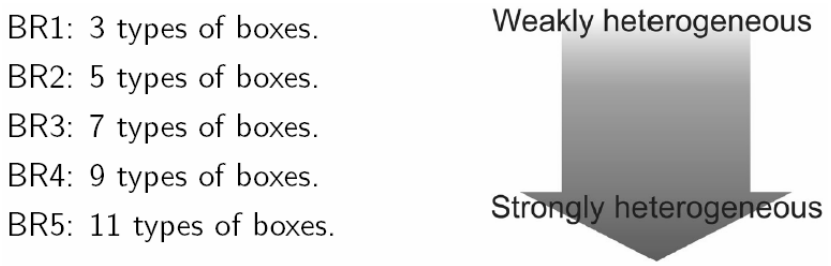

Fig. 4 The problem libraries used in our implementation.

We compare the average of use of the container space of our algorithm, cf. presented by Araújo and Pinheiro (2010) (represented in Table 2 by HBGA) with: a genetic algorithm proposed by Gehring and Bortfeldt (1997) (GA), the constructive algorithm on Bischoff and Ratcliff (1995) (CA), a constructive algorithm by Bischoff, Janetz and Ratcliff (1995) (HBal), the hybrid genetic algorithm described by Bortfeldt and Gehring (2001) (HGA), the parallel genetic algorithm by Gehring and Bortfeldt (2002) (PGA), a proposed heuristic by Bischoff (2006) (PH) and Tabu Search proposed in Bortfeldt, Gehring, and Mack (2002) (TS).

We present the average of use of the container space of our algorithm and its standard deviation (σ). We have limited the execution time to 10 minutes (execution timeout parameter).

Table 2 Comparing some proposals.

File	GA	CA	HBal	HGA	PGA	PH	TS	**HBGA**	
								%	σ
BR1	86.77	83.37	81.76	87.81	88.10	89.39	93.23	**92.13**	2.76
BR2	88.12	83.57	81.70	89.40	89.56	90.26	93.27	**91.09**	3.15
BR3	88.87	83.59	82.98	90.48	90.77	91.08	92.86	90.38	2.72
BR4	88.68	84.16	82.60	90.63	91.03	90.90	92.40	89.07	3.02
BR5	88.78	83.89	82.76	90.73	91.23	91.05	91.61	88.17	3.44

We observe better results, comparing with most other approaches, for homogeneous and weakly heterogeneous problems described in the OR-Library instances. These results will still be improved by extending the timeout parameter although the high computational cost as time and resources.

Our application automatically generates a MATLAB file which shows the packing result for each problem. Fig. 4 presents the solution for the first problem in BR1 library.

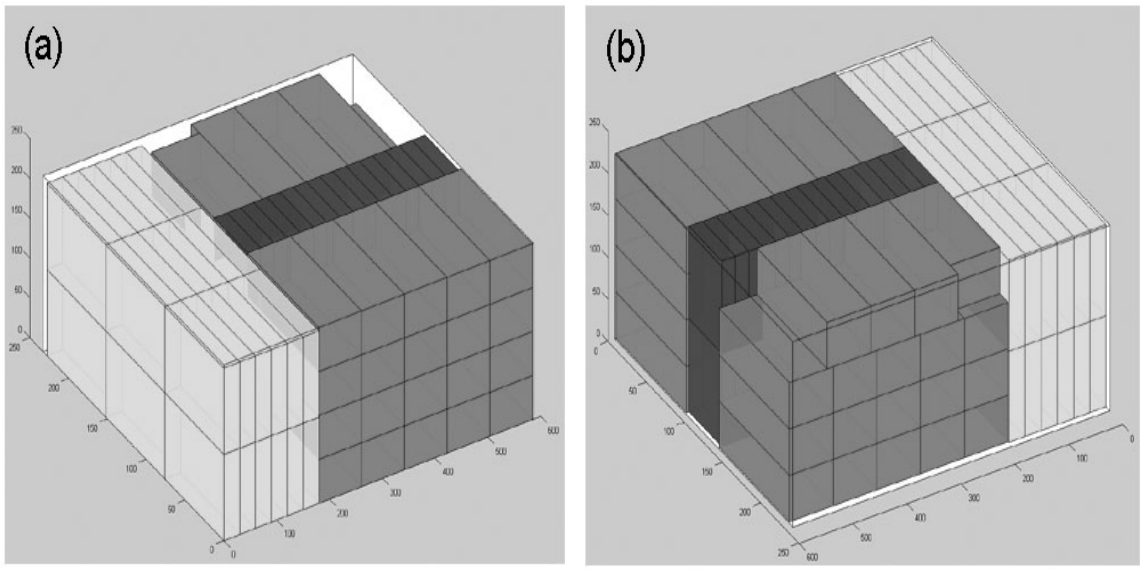

Fig. 5 A packing solution: (a) front view and (b) back view.

Case study ESMALTEC is a highly regarded Brazilian company, founded on 1963, that produces stoves, refrigerators, freezers and water coolers. It is one of the most modern factories in Latin America, and sells its products in over 50 countries.

The Containers are used to transport the many type of products. Real data about the most common cases were used by Nepomuceno, Pinheiro and Coelho (2007) to compare the company's results with results of the Hybrid Algorithm (HA). We add a work, Davies and Bischoff (1999), that lists some results of problems with a single type of box (not the same instances of problems that in this work). The work cites Gehring, Menschner, and Meyer (1990) (represented by GMM in Fig. 5), the layering approach (LA) and the column building procedure of Bischoff and Ratcliff (1995) (CB). Our proposal is represented by HBGA.

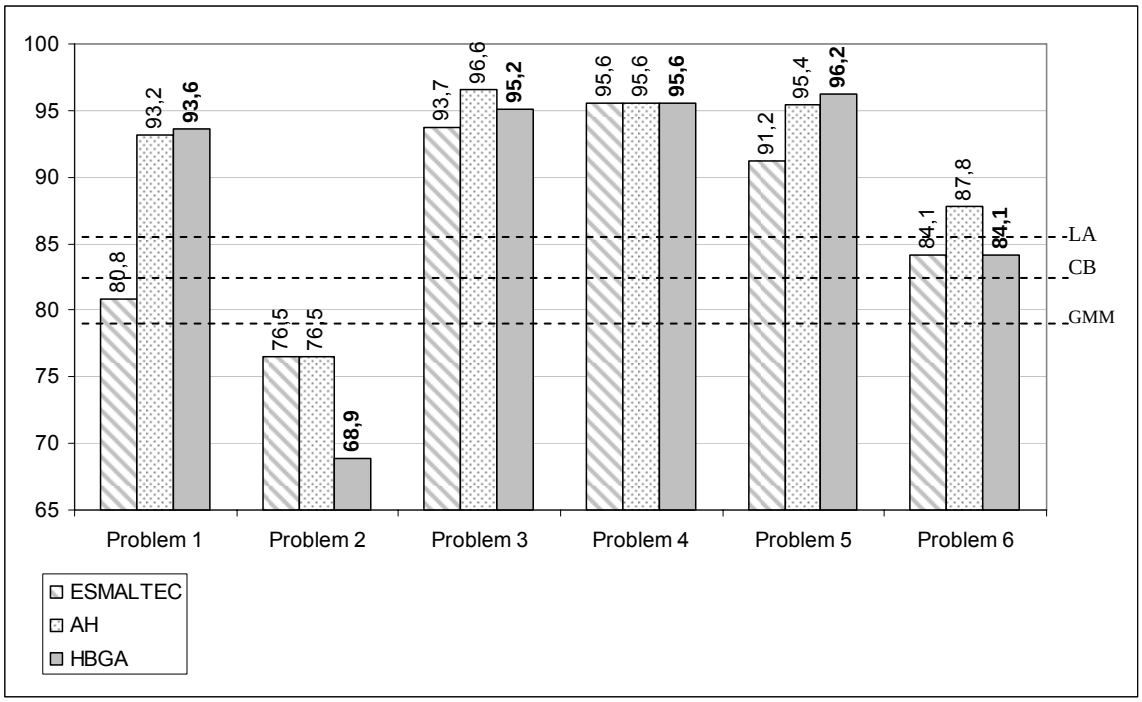

Fig. 6 Comparing results of the works.

We can notice that, for all problems used, our results were better or equals to those by factory's system. Fig. 6 shows the solution for the problem 5 (best result).

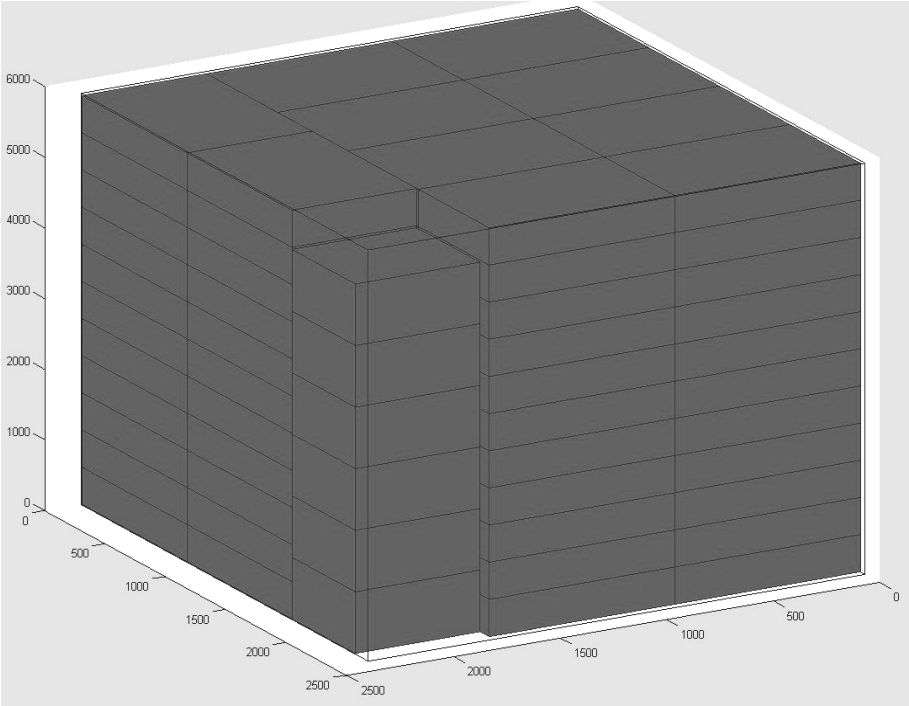

Fig. 7 The found solution for the problem 5.

We emphasize the low average running time, about 45.17 seconds.

5. Final Remarks

The current algorithm finds good solutions in order to maximize the loaded volume. These results, in most cases, are better than results from others approaches to the same library used on tests.

We hope to improve the execution time by changing the backtracking method by another search algorithm as well to do more computational tests using other test suites.

We also intend to implement some heuristics on non-linear programming model in order to not use parameters for most the heuristics.

Finally, we expect to use the implemented algorithm and its results in our case study, suggesting better results to ESMALTEC Company.

Acknowledgements The second author acknowledges the financial support received from the Brazilian Counsel for Technological and Scientific Development (CNPq).

References

Araújo, L.J.P., Pinheiro, P.R., Combining Heuristics Backtracking and Genetic Algorithm to Solve the Container Loading Problem with Weight Distribution, Advances in Soft Computing, (73) pp. 95-102. (2010)

Beasley, J.E., OR-Library: distributing test problems by electronic mail. Journal of the Operational Research Society 41(11) pp. 1069-1072. (1990)

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

Bischoff, E.E., Janetz, F., Ratcliff, M.S.W., Loading Pallets with Nonidentical Items. European Journal of Operational Research, 84:681-692. (1995)

Bischoff, E.E., Ratcliff, M.S.W., Issues in the Development of Approaches to Container Loading. Omega 23:377-390. (1995)

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

Bortfeldt, A., Gehring, H., Mack, D., A Parallel Tabu Search Algorithm for Solving the Container Loading Problem, Parallel Computing 29, 641-662. (2002)

Chen, C.S., Lee, S.M., Shen, Q.S., An analytical model for the container loading problem. European Journal of Operations Research, 80:6876. (1993)

Davies, A.P., Bischoff, E.E., Weight distribution considerations in container loading. European Journal of Operations Research, 114:509-527. (1999)

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

Gehring, H., Bortfeldt, A., A Genetic Algorithm for Solving the Container Loading Problem. Internat. Trans. in Operational Research. 4:401-418. (1997)

Gehring, H., Bortfeldt, A., A Parallel Genetic Algorithm for Solving the Container Loading Problem, International Transactions in Operational Research, 9:497-511. (2002)

Gehring, H., Menschner, K., Meyer, M., A computer-based heuristic for packing pooled shipment containers. European Journal of Operational Research, 44:277-288. (1990)

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

Mack, D., Bortfeldt, A., Gehring, H., A parallel hybrid local search algorihtm for the container loading problem. International Transactions in Operations Research, 11:511-533. (2004)

Martello, S., Pisinger, D., Vigo, D., The three-dimensional bin packing problem. Operational Research, 48:256-267. (2000)

Morabito, R., Arenales, M., An and/or-graph approach to the container loading problem. International Transactions in Operational Research, 1(1), 59-73. (1994)

Nepomuceno, N., Pinheiro, P.R., Coelho, A.L.V., Tackling the Container Loading Problem: A Hybrid Approach Based on Integer Linear Programming and Genetic Algorithms. In: VII European Conference on Evolutionary Computation in Combinatorial Optimization (EVOCOP). Berlin: Springer, 2007. v.4446. p.154 - 165. (2007)

Pisinger, D., Heuristc for the Conteiner Loading Problem. European Journal of Operational Research, 141:382-392. (2000)

Pisinger, D., Heuristics for the Container Loading Problem. European Journal of Operational Research, 141:143-153. (2002)

Wascher, G., Hausner, H., Schumann, H., An improved typology of cutting and packing problems. European Journal of Operational Research, Vol. 183, No.3. pp.1109-1130. (2007)