Container loading for nonorthogonal objects with stability and load bearing strength compliance

Blind for submission

***Abstract*—A feasible solution for a container loading problem is the exact order, position and orientation in which the objects are loaded into the container. However, the more constraints are taken into account, the more complex the problem is. Several algorithms have been proposed to load objects with different shapes and taking into account a few constraints. Nevertheless, most of the approaches restrict the shape to be orthogonal (i.e., standard boxes) and constraints like stability or load bearing strength were not considered.**

The aim of this paper is to provide an approach to solve the problem of loading orthogonal and non-orthogonal boxes, considering them as polyhedral. In addition, our proposal deals with constraints on the dimensions of the container, whether every box can be rotated in any of the three dimensions, the maximum load bearing strength and also the minimum stability required. To solve the problem with the characteristics mentioned above a data structure is proposed to represent the i-th state of the container's loading process by storing the height values and the remaining load bearing strength of the objects inside the container.

I. Introduction

Container loading is a significant area for companies engaged in the transportation of goods and logistics in which the optimization of the resources play an important role.

Container loading problems can be formalized by defining the properties of the container and the objects. Also there is the need to take into account some constraints that determines whether the final solution is acceptable. After these definitions the container loading problem becomes a constrained optimization problem whose function to maximize is the container's space allocated.

Getting an optimal solution is almost unreachable because they are NP-hard problems [9]. Due to this, heuristic methods have to be considered. Those methods usually are specialized by loading the same kind of boxes. This paper deals with the generalization of those methods and the generalization will be done by enabling the load of standard boxes, dodecahedron and hexagonal prisms.

This paper extends [16] by introducing a new heuristic approach that loads the container by evaluating a set of criteria. This is in contrast of the vertex management approach used in e.g. [13], [16]. In addition, while we previously used a local or global search, we consider here an exploration of the weight's space.

We will assess the performance of our approach when dealing with orthogonal boxes by solving the benchmark created by Bischoff and Ratcliff [3]. Later on, we will present the performance of our approach applied to non-orthogonal objects running our proposed benchmark for polyhedral boxes.

The structure of this paper is as follows. In Section 2, we describe step by step the heuristic described in [2] to load standard boxes and its extension for nonorthogonal boxes while giving a detailed overview of the criteria used for box selection. In Section 3, we explain how to deal with the problem of getting feasible spaces where to load each box and how to aggregate the criteria. In Section 4, we describe the experiments we have performed and the results obtained. This paper finishes with conclusions and future work.

II. Nonorthogonal objects: the heuristic

In this paper we propose an extension of the method proposed by E.E. Bischoff [2] to deal also with nonorthogonal boxes, in particular, dodecahedra and prisms. The container is loaded box by box, applying a heuristic to assess the suitability of the different options for where and how each box could be loaded. Formally, all boxes to be allocated are represented as an array of boxes ordered by volume in decreasing order. Then, at a given point, when a new box is needed, a box is selected on the basis of five criteria. Each criterion is considered for each box, each possible box rotation and for each feasible location.

In general, the following steps can be distinguished in the approach:

1. **Identify all fully supported spaces**
2. **Identify all feasible placements for single boxes**
3. **Evaluate the box-space combinations found**
4. **Select combination with the highest score and place the box**
5. **Update relevant parameters**

We give below a description of each of these steps. Furthermore, we describe in detail the third step, the one used to evaluate the box-space combinations found considering the five criteria.

1) Identify all fully supported spaces: The first step to allocate a box into the container is to identify the available empty spaces which can be considered as candidates for containing the box. When the box can be rotated, different rotations might lead to different sets of possible empty spaces. So, we have to search for the empty spaces of each possible rotation.

The occupation of the container is supported with a 2D data structure representing container's length and width. The value of each single position of the matrix represents an empty position in case of a 0 and the height of a occupied space in case of a value greater than 0. Details on how to maintain

this matrix updated after loading a box, and also on how to identify the available spaces are described in Section 3.

Furthermore, to finally consider a space able to fit a box, it has to satisfy a maximum load bearing capacity constraint. In this way, another 2D data structure is needed to maintain updated the remaining load bearing strength of every single space. The aim of this second data structure is to ensure the satisfaction of the maximum load bearing strength but the operation is almost the same as the one used to identify all possible empty spaces.

2) Identify all feasible placements for single boxes: The next step is to filter all the candidate spaces found, and keep only those that can contain the box. Part of this step can be done while identifying all fully supported spaces. For example, we can filter the possible box-space combinations found in the previous step by removing the spaces with height values (basement height + box height) larger than the height of the container. Another filter is to remove the box-space combinations not able to fit the box in process.

After applying the filters we are ready to calculate for all the box-space combinations found the criteria. This is described in the next step.

3) Evaluate the box-space combinations found: E.E. Bischoff in [2] proposed five criteria to evaluate the different box-space combinations. We describe their rationale and our adapted criteria which are based on the ones of Bischoff but taking into account that the boxes to be loaded can be nonorthogonal.

Criterion 1 (+): Relation between box size and position of loading surface. This criterion supports loading big boxes on the bottom so we may have opportunities to load smaller boxes on them. We have applied this criterion as suggested by E.E. Bischoff. In our case, we consider the difference between the height of the container H and the vertical position where the space is located S_H. This difference is then multiplied by the result of the box volume V_B.

$$C_1 = (H - S_H)V_B$$

Criterion 2 (+): Match between box and space dimensions. This second criterion assesses the similarity between a box and the space dimensions. Our approach changes the implementation of this criterion but maintains its rationale. If S_A means the amount of available space in this region and S_N denotes the needed space to load on this region the box in process, then we can write the formula of the criterion as follows.

$$C_2 = 1 - \frac{S_A - S_N}{S_A}$$

This criterion will reach 1 when $S_A = S_N$, maximum similarity; and 0 when $S_N = 0$, minimum similarity; and values in the range (0,1) otherwise.

Criterion 3 (-): Unusable space generated. Unlike the previous criteria this one penalizes the difference between box and space dimensions because it can generate new unusable spaces after fitting the box in process. In some sense, this criterion is the opposite of the previous one but taking into account if the type of the box in process is a prism, a dodecahedron or a simple orthogonal box. Hence this criterion will be weighted with a bigger value in case of dodecahedron ($\alpha = 1.4$) because of the fact that this kind of box generates more unusable spaces than a prism or an orthogonal box. In the same way, in the case of prisms ($\alpha = 1.2$) the weight will be bigger than the one in the case of orthogonal boxes ($\alpha = 1$).

$$C_3 = \alpha(S_A - S_N)$$

Criterion 4 (+): Potential for building column of identical boxes. This criterion supports the cases that maximize the similarity between box and space dimensions and minimize the unusable space generated while remaining boxes of the same type. The only modification we have done in this criterion is to use the volume of the different kind of boxes. In this formula $m_i^{'}$ means the number of boxes of the same type as the actual one that remains unloaded. H, S_H and V_B has the same meaning than in the first criterion and d_{i1}, d_{i2} and d_{i3} represents the box's length, width and height. S_L and S_w takes the value of the container's length and width. This criterion also takes into account the minimum load bearing capacity on the top surface B_{top} on which the actual box will be placed. Its value is calculated as the minimum between b_{i3} (the maximum load bearing on the top of the actual box) and $B_{min} - \frac{w_i}{d_{i1}d_{i2}}$ (the difference between the minimum spaces remaining load bearing and the loss of load bearing capacity will be generated in case we load the actual box).

$$C_4 = min\{m_i^{'}, \tfrac{H-S_H}{d_{i3}}, 1 + \frac{B_{top}d_{i1}d_{i2}}{w_i}\} \times \frac{V_B}{S_L S_w}$$

$$B_{top} = min\{b_{i3}, B_{min} - \frac{w_i}{d_{i1}d_{i2}}\}$$

Criterion 5 (-): Relative loss in load bearing capacity. This criterion adversely affects the total score of the box-space combination in case we take into account the load bearing capacity of the different box types. A big value of this criterion means that we are trying to load a heavy box on the top of another one decreasing its load bearing capacity and causing a strong impact on subsequent placement opportunities. With this criterion we want to reward loading heavy boxes in low altitudes, the best reward is given when the heavy box is placed on ground levels.

$$C_5 = (B_{avg} - B_{top}d_{i1}d_{i2}) \times \tfrac{H-S_H-d_{i3}}{d_{i3}}$$

This formula will be applied only if C_5 meets the following inequation:

$$\tfrac{B_{top}}{H-S_H-d_{i3}} < D_{max}$$

Otherwise $C_5 = 0$. Hence with this criterion we only penalize when after loading the box, we surely know that the box with largest density cannot be placed in the future above the loaded box.

4) Select the combination with the highest score and place the box: Once we have the value of the five criteria for

each feasible box-space combination, the overall score for each combination is calculated as a weighted sum of the five criteria. In Section 3, we explain how to aggregate the five criteria as an overall score using a weighted sum.

5) Update relevant parameters: This final step depends on the data structure used to represent the boxes inside the container. The data structures used in our approach are two 2D matrices. Then, a modified version of the flood fill scan line algorithm [22] give us the spaces available in the matrix and also updates the container's spaces filling them with the new height values (basement height + box height). To update the data structure representing the remaining load bearing strength, we consider for every single space (every space represents the shape of the box in process) the difference between the remaining load bearing strength and the loss caused by the load of the box in process.

III. Nonorthogonal objects: Solving the problem

In this Section we describe in more detail some of the steps exposed above. In particular, we discuss how to identify all fully supported spaces, how to handle the stability and load bearing strength constraints and how to calculate the overall score for every box-space combination to select the one with highest score to load the current box. Also, we will describe how we ensure the stability and maximum load bearing strength compliance.

We start describing how to keep up to date the 2D matrix data structure we have used to identify all free spaces, the algorithm applied to obtain the candidate spaces to load the box in process, and how we applied filters to identify all feasible placements of single boxes. Later on, we will describe the second 2D data structure used to control the load bearing ability and, afterwards, we will show how we adapted the criteria and how we used them to calculate the overall score.

A. Getting potential spaces where to load the box

Two methods have been used to get the box-space combinations but both use the same data structure. A 2D matrix, $length \times width$ of the container, contains the up to date height of all boxes already loaded. Each space found by these methods is described by its area and starting position X, Y and Z. The area is used to filter the spaces according to the free space needed to load the box. The spaces' precision is in centimeters because the container's dimensions are expressed in centimeters.

The first method gets the spaces looking on the entire 2D matrix for spaces of the same height. When they are found, the space is marked as visited and the starting coordinate found is defined as its position, and the amount of marked coordinates is defined as its area. The search is done along the x-axis and the y-axis. One of the disadvantages of this method is the value assigned to the starting position of the space because when used to load a box might cause a collision. I.e., a space with a big area can produce a collision when loading the box because the position is under a dodecahedron. To prevent this

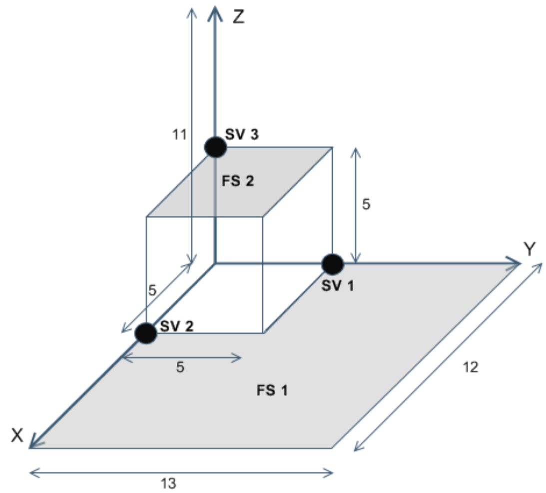

Fig. 1. Snapshot of an hypothetical second step of the container loading process after one box has been loaded and then two feasible spaces where to load the following box are found (in different grey tones).

5	5	5	5	5	0	0	0	0	0	0	0	0
5	5	5	5	5	0	0	0	0	0	0	0	0
5	5	5	5	5	0	0	0	0	0	0	0	0
5	5	5	5	5	0	0	0	0	0	0	0	0
5	5	5	5	5	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0

Fig. 2. 2-D matrix structure values for the example in Figure 1. The two feasible spaces are represented by the value 5 and 0 corresponding to the height of every single space.

type of problem, we require the space to have almost the same length and width than the box.

The second method proposed is based on the first one. In this case, however, the starting position of the space is a vertex. Formally, the vertex management approach [13] uses a list of vertexes where objects can be loaded. The first vertex is $(0, 0, 0)$ and after a box is loaded the corresponding vertex is removed from the list adding some new vertexes. For all box's types three new vertexes are added, $(x_i + w_i, y_i, z_i)$, $(x_i, y_i + d_i, z_i)$ and $(x_i, y_i, z_i + h_i)$, where (x_i, y_i, z_i) states for the reference position of the box b_i and w_i, d_i, h_i represents its dimensions. In case of a nonorthogonal box some heuristically selected vertexes are added too in order to allow the possibility to load small boxes below a dodecahedron or to form honeycomb-like structures with prisms. This second approach obtains spaces with less probability of collision while loading.

The procedure to maintain the heights up to date in the 2D matrix is applied after every box b_i is loaded. The input of

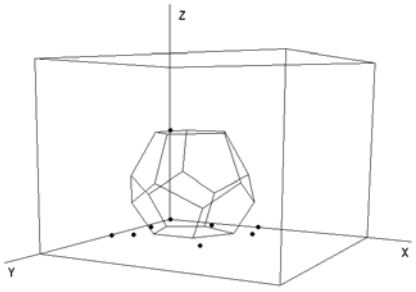

Fig. 3. Dodecahedron with the corresponding vertexes for the vertex management approach used in the second method for getting potential spaces where to load the box.

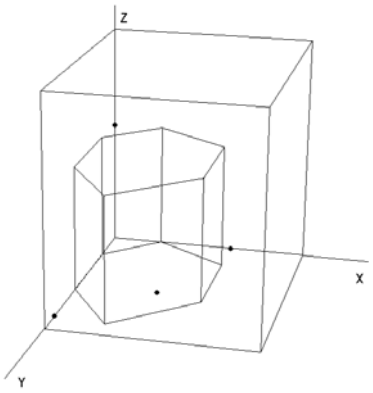

Fig. 4. Hexagonal prism with the corresponding vertexes for the vertex management approach used in the second method for getting potential spaces where to load the box.

this procedure is the box's position, length and width. The output of the procedure is the height for all coordinates (i, j) for $i = x_i$ to $x_i + d_i$ and $j = y_i$ to $y_i + w_i$. Figure 2 represents the updated state of the 2D matrix once the box in Figure 1 has been loaded.

B. Handling the load bearing strength constraint

The method proposed to support the maximum load bearing strength associated to each dimension of a box is based in another 2D data structure similar to the one used to get the potential spaces where to load the box. The same procedure is applied after every single box is loaded inside the container and it takes into account the remaining load bearing strength of the space before the box was loaded, the box's weight and the box's load bearing strength. Hence, the input of this procedure is the box's position, length and width and also the box's load bearing capacity and weight. The output of the procedure is the remaining load bearing strength after the box has been loaded or the box's load bearing strength in case the former is bigger than the latter. If the cells with the value 5 in Figure 2 are replaced by the remaining box's load bearing strength then we get the updated state of the 2D data structure that supports the load bearing constraint.

C. Handling the stability constraint

To ensure the compliance of the stability constraint, we use the two dimensional data structure used to get potential spaces where to load the box. This is an easy process where we have to ensure that the height values corresponding to the space where to load the box are most of them equal to the box's starting at coordinate z_i. The percentage of equal values can be used to ensure a degree of stability. This is to say, if we ensure a 100% stability compliance all the height values corresponding to the space where to load the box have to be equal to z_i.

D. Criteria: overall score

Taking into account the criteria above, the following score is computed for each box:

$$E(box) = v_1 C'_1(box) + v_2 C'_2(box) + v_3 C'_3(box) + v_4 C'_4(box) + v_5 C'_5(box)$$

In this definition, we have assumed that all criteria, including criteria 3 and 5, are positive and that their values are bounded (they belong to the unit interval). In addition, we require the weights v_i add to one (i.e., $\sum_{i=1}^{5} v_i = 1$). In this way, the expression is a weighted mean of the criteria. Formally, $C'_i = C_i / r_i$ where $r_i = max_{k=1}^{n} r_k$ for $i = 1, 2$, and 4, and $C'_i = C_i / r_i$ where $r_i = min_{k=1}^{n} r_k$ for $i = 3$ and 5.

E. Procedures to find out the best weights v_i

The score given above is used for selecting which of the available boxes should be loaded next in the container. Nevertheless, the score depends on the weights v_i. We have studied a few different approaches to define the weights v_i. We describe them briefly below.

Note that the goal is to obtain weights, that in average, result into a good volume of occupation for all the problems in the benchmarks.

The first procedure used to investigate the effect of the different weights while loading the container is the pure random search. This procedure turned out not very useful because the volume occupation in case of very similar weights became the same, and we had very similar values for different generated random weights. Another approach was to apply the Simplex method of Nelder and Mead [14]. Again this procedure turned out unusable for the same reason. Due to these bad results, we studied the behavior of a large set of possible weights values in some representative problems and then extrapolate to all the problems in the benchmark.

In particular, we have considered all weights in the set

$$\{(v_1, \ldots, v_5) | v_i \in \{0, 0.25, 0.5, 0.75, 1\} and \sum_{i=1}^{5} v_i = 1\}.$$

The results obtained are analysed in Section 4.

```
587    233    220
10       29.3680757000
108  0    76  0    30  1   20   246.24  0.02824  0.16575  0.11365
110  0    43  1    25  1   11   118.25  0.26897  0.13148  0.17733
 92  1    81  1    55  1    9   409.86  0.00902  0.33191  0.25007
 81  0    33  1    28  1    9    74.84  0.24737  0.27825  0.21347
120  1    99  1    73  1    9   867.24  0.03681  0.25852  0.14228
111  0    70  1    48  1   13   372.96  0.26002  0.14047  0.32586
 98  0    72  1    46  1    4   324.58  0.07571  0.29199  0.26801
 95  0    66  0    31  1   12   194.37  0.09286  0.06752  0.08617
 85  0    84  0    30  1    9   214.20  0.34896  0.23626  0.35238
 71  0    32  1    25  1   10    56.80  0.01198  0.15825  0.32711
```

Fig. 5. First problem in the wtpack4 problem's file. It corresponds to a container of dimensions $(587, 233, 220)$ and boxes of 10 different shapes.

IV. EXPERIMENTS

In this section we describe the experiments performed and analyze the results. Two sets of experiments have been considered. The weights used in the experiments are listed below. Later on, we will use w_1 to mean weights 1 an so on:

- weights 1 = [0.5, 0, 0.5, 0, 0]
- weights 2 = [0.078445, 0.338914, 0.108992, 0.210489, 0.2631581]
- weights 3 = [0.254426, 0.016283, 0.154418, 0.328976, 0.245895]

A. Bischoff and Ratcliff files

To compare our approach to previous works dealing only with orthogonal boxes we have considered the problems described in [3] and accessible through the web site in [19]. They consist of 7 sets of 100 problems. In each set, all problems have the same number of different boxes, although the sizes of the boxes are not the same. Figure 5 represents one of the problems from the set of problems wtpack4 (10 different boxes's shapes). In the figure, the first row corresponds to the dimensions of the container, the second row to the number of different boxes and the total volume, the other rows include the pairs (dimension, vertical position?) – the latter being a boolean value 0/1 indicating whether the dimension can be put in vertical position-, the number of boxes, and additional information about the boxes (weight and maximum load on the corresponding surface). Test files can be found in [21].

Table I shows the results when applying the second method described in Section III-A for getting the potential spaces where to load the box. The results we have obtained when just considering the stability constraint (LBA OFF) are slightly worst than the ones obtained by Bischoff and Ratcliff. A difference of about 5% or less is obtained. Nevertheless, difference is at the cost of permitting additional shapes for the boxes. In any case, for some of the particular problems we achieve a good score e.g.. 87.51% or 89.23%.

To show the impact of the load bearing ability constraint on the container loading process, Table I also shows the results when considering five different degrees of load bearing ability LBA. Starting from a degree of load bearing strength, (LBA TEST 1), and decreasing the load bearing strength down to (LBA TEST 5). The results shows that the relation between the volume occupied is directly proportional to the average time needed to load all the containers in each wtpack set of problems. That is, the smaller the average volume occupied, the smaller the average computational time. This is because when the load bearing constraint is less permissive, the number of boxes that are able to be loaded are less. So the main impact on the average time needed to load the containers is proportional to the time spent to evaluate the box-spaces combinations and to detect the collisions. In contrast, it is not so relevant the 2D data structure management process to deal with the load bearing ability.

		wtpack1	wtpack2	wtpack3	wtpack4	wtpack5	wtpack6	wtpack7
Avg. Occup. (%)	LBA OFF	75.41	75.49	75.94	75.40	75.74	75.52	75.36
	LBA TEST 1	63,61	63,68	63,80	63,63	63,60	63,43	63,60
	LBA TEST 2	59.10	58.90	58.66	59.15	59.08	59.33	59.02
	LBA TEST 3	57.66	57.27	58.02	57.30	58.00	57.99	57.82
	LBA TEST 4	56.11	56.31	56.15	56.23	56.12	56.30	56.03
	LBA TEST 5	54.15	54.18	53.87	54.01	53.93	53.95	53.67
Max. Occup. (%)	LBA OFF	86.36	87.51	87.44	86.77	86.95	86.77	89.23
	LBA TEST 1	85,20	84,81	85,47	84,23	85,07	84,76	86,00
	LBA TEST 2	85.20	85.60	85.60	84.94	84.81	86.14	85.03
	LBA TEST 3	85.60	86.00	84.94	85.56	84.03	85.74	85.34
	LBA TEST 4	85.60	86.18	84.81	85.07	85.38	84.23	85.20
	LBA TEST 5	83.06	83.32	83.46	82.52	83.72	84.03	83.32
St. Dev. (%)	LBA OFF	06.04	06.26	06.00	05.99	05.77	05.73	06.68
	LBA TEST 1	17,57	17,24	17,05	17,60	17,36	17,66	17,67
	LBA TEST 2	19.84	20.03	19.81	20.25	20.19	19.89	19.88
	LBA TEST 3	20.50	20.46	20.48	20.46	20.39	19.87	20.45
	LBA TEST 4	21.13	21.12	20.95	21.43	20.96	21.17	21.16
	LBA TEST 5	22.41	22.36	22.03	22.11	22.22	22.04	22.06
Avg. Time (sec.)	LBA OFF	24.86	24.03	24.17	25.44	24.84	26.06	25.06
	LBA TEST 1	24,49	24,07	24,00	23,39	23,64	23,66	23,55
	LBA TEST 2	21.66	18.38	18.75	21.69	22.68	22.07	19.04
	LBA TEST 3	20.07	17.63	18.19	18.50	19.34	21.74	21.97
	LBA TEST 4	17.74	16.71	17.43	17.01	17.75	16.78	16.86
	LBA TEST 5	16.29	15.94	15.67	16.25	16.23	16.31	16.45

TABLE I
AVERAGE OCCUPATION, MAXIMUM OCCUPATION, STANDARD DEVIATION AND AVERAGE EXECUTION TIME WHEN PROCESSING THE BISCHOFF AND RATCLIFF FILES WITHOUT CONSIDERING THE LOAD BEARING ABILITY (LBA OFF) AND WHEN CONSIDERING IT WITH DECREASING DEGREES OF LOAD BEARING ABILITY (LBA TEST 1. . . 5).

Case	Boxes	Dodec.	Hex. prism	First Method. w1-w2-w3	Second Method. w4-w5-w6
Case 1	100%				
Case 2	50%		50%	60.14-60.01-60.93	57.97-58.18-58.2
Case 3	50%	50%		48.00 -47.70-48.02	51.09-50.80-50.96
Case 4			100%	51.14-53.83-51.74	47.40-48.56-47.85
Case 5		50%	50%	42.21-42.23-41.49	42.73-43.33-42.53
Case 6		100%		34.96-35.00-34.72	39.38-40-39.46
Case 7	33%	33%	33%	48.22-48.96-48.31	43.34-44.46-43.86

TABLE II
DESCRIPTION OF OUR VARIATION OF BISCHOFF AND RATCLIFF FILES CONSIDERING LOAD BEARING STRENGTH, AND MEAN OCCUPANCY ACCORDING TO BOX TYPES, METHOD USED AND WEIGHTS APPLIED.

Figure 6 and Figure 7 depicts these results. A strange effect of the different degrees of LBA is present in Figure 7. While the average volume occupied remains almost constant from wtpack1 to wtpack5 for all tests, the average time varies more in some tests rather than others. This can be explained with the fact that each $wtpack_i$ has different boxes' sizes and the same volume can be occupied with different combinations of boxes, e.g. either loading three little boxes or loading one big box with triple in size. The former may need more time to load the three boxes than the latter. And the degree of load bearing ability is applied to each box's type so we won't have any problem loading the big box but we may not be able to load the boxes one above the other because the remaining load bearing strength.

B. Variation of Bischoff and Ratcliff files

This is a variation of the files described in the previous section, where other kinds of boxes are also permitted. We have considered, as in the Bischoff and Ratcliff files, sets of 3, 5, 8, 10, 12, 15 and 20 different box types. Then, for each

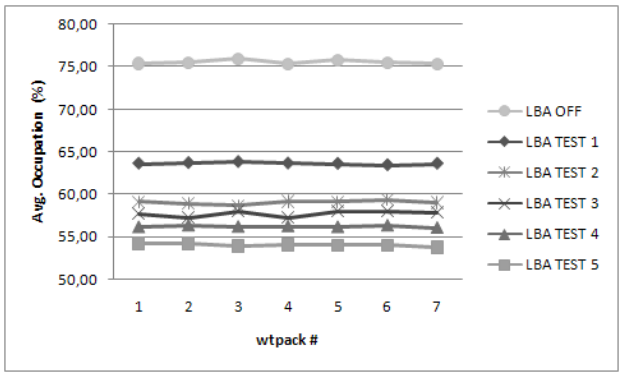

Fig. 6. Average occupation when processing the Bischoff and Ratcliff files without considering the load bearing ability (LBA OFF) and when considering it with decreasing degrees of load bearing ability (LBA TEST 1...5).

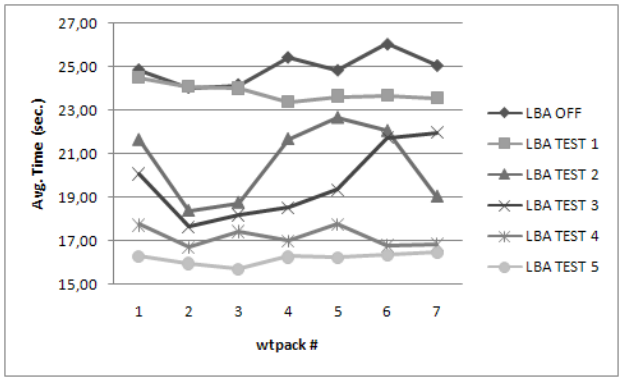

Fig. 7. Average processing time when processing the Bischoff and Ratcliff files without considering the load bearing ability (LBA OFF) and when considering it with decreasing degrees of load bearing ability (LBA TEST 1...5).

of these sets of boxes, we have considered different cases corresponding to different proportions of boxes of different shapes. Table II gives the 7 cases considered. That is, we have considered the case of only boxes (as in Bischoff and Ratcliff files), the case of 50% regular orthogonal boxes and 50% hexagonal prisms, the case of 50% regular orthogonal boxes and 50% dodecahedron, etc. The data files are publicly available trough the web page given in [21].

As expected, the results obtained are worse than the ones when only orthogonal boxes were considered. For example, the worst case is when only dodecahedra are considered. However, this is natural because its shape leaves empty spaces that cannot be filled with standard size boxes.

V. Conclusions and future work

In this paper we have considered the container loading problem for nonorthogonal objects. Our approach has been applied to three types of objects: typical boxes (orthogonal boxes), dodecahedra and hexagonal prisms. We have presented the results of our approach that are similar to the ones described in Bischoff and Ratcliff but permitting its application to the new types of objects considered and taking into account both the stability and maximum load bearing ability constraints.

As future work we consider the extension of the approach to take into account a correct distribution of the weight and also increase the mean occupancy when considering dodecahedra and hexagonal prisms.

VI. Acknowledgments

This work is partially supported by the Spanish Ministerio de Fomento (project PON, T27/2006) and MEC (CONSOLIDER INGENIO 2010 CSD2007-00004, and TSI2007-65406-C03-02) and by the Generalitat de Catalunya (AGAUR, 2006BE-2 00338).

References

[1] Birgin, E. G., Martínez, J. M., Ronconi, D. P. (2005) Optimizing the packing of cylinders into a rectangular container: A nonlinear approach, European Journal of Operational Research, 160, 19-33.
[2] Bischoff, E. E. (2006) Three-dimensional packing of items with limited load bearing strength, European Journal of Operational Research, 168, 952-966.
[3] Bischoff, E. E., Ratcliff, M. S. W. (1995) Issues in the Development of Approaches to Container Loading, Omega Int. J. of Management Sciences, 23:4 377-390.
[4] Bortfeldt, A., Gehring, H. (2001) A hybrid genetic algorithm for the container loading problem, European Journal of Operational Research, 131, 143-161.
[5] Bortfeldt, A., Gehring, H., Mack, D. (2003) A parallel tabu search algorithm for solving the container loading problem, Parallel Computing, 29 641-662.
[6] Davies, A. P., Bischoff, E. E. (1999) Weight distribution considerations in container loading, European Journal of Operational Research, 114, 509-527.
[7] Dyckhoff, H. (1990) A typology of cutting and packing problems, European Journal of Operational Research, 44, 145-159.
[8] Dyckhoff, H., Finke, U. (1992) Cutting and Packing in Production and Distribution, Physica, Heidelberg.
[9] Eley, M. (2002) Solving container loading problems by block arrangement, European Journal of Operational Research, 141, 393-409.
[10] Eley, M. (2003) A bottleneck assignment approach to the multiple container loading problem, OR Spectrum, 25 45-60.
[11] Gehring, H., Bortfeldt, A. (1997) A Genetic Algorithm for Solving the Container Loading Problem, Int. Trans. Operational Research, 4:5/6, 401-418.
[12] George, J. A., George, J. M., Lamar, B. W. (1995) Packing different-sized circles into a rectangular container, European Journal of Operational Research, 84, 693-712.
[13] Miyamoto, S., Endo, Y., Hanzawa, K., Hamasuna, Y. (2006) Metaheuristic Algorithms for Container Loading Problems: Framework and Knowledge Utilization, J. of Advanced Intelligence and Intelligent Informatics, 11:5 51-60.
[14] Nelder, J. A., Mead, R. (1965) A simplex method for function minimization, Computer Journal, 7, 308-313.
[15] Pisinger, D. (2002) Heuristics for the container loading problem, European Journal of Operational Research, 141, 382-392.
[16] Torra, V., Cano, I. Miyamoto, S., Endo, Y. (2008) Container loading for nonorthogonal objects using local search and simulated annealing, Submitted.
[17] Zachmann, G. (1994) Exact and Fast Collision Detection, Diploma Thesis, Technischen Universitt Darmstadt.
[18] Zachmann, G. (2000) Virtual Reality in Assembly Simulation Collision Detection, Simulation Algorithms, and Interaction Techniques, PhD Dissertation, Technischen Universitt Darmstadt.
[19] http://people.brunel.ac.uk/˜mastjjb/jeb/orlib/files/wtpack1.txt (to wtpack7.txt)
[20] http://www.cs.unc.edu/˜geom/collide/
[21] http://www.iiia.csic.es/˜vtorra/projecte.pon/
[22] http://www.student.kuleuven.be/˜m0216922/CG/floodfill.html