Applied Automatic Systems: Proceedings of Selected AAS 2009 Papers, Ohrid 26-29.09.2009
Published by the Society for ETAI of Republic of Macedonia, Skopje, 2009

An Efficient Algorithm for 3D Rectangular Box Packing

M.Zahid Gürbüz *, Selim Akyokuş, İbrahim Emiroğlu***, Aysun Güran******

**Yıldız Technical University, İstanbul, 34210 TR(Tel: 505-483-4699; e-mail: zgurbuz@dogus.edu.tr).*
***DogusUniversity, İstanbul, 34722 TR (e-mail:sakyokus@dogus.edu.tr)*
****Yıldız Technical University, İstanbul, 34210 TR(Tel: 212-383-4599; e-mail: emir@yildiz.edu.tr).*
*****Yıldız Technical University, İstanbul, 34210 TR(Tel: 536-947-3275; e-mail: adogrusoz@dogus.edu.tr).*

Abstract: Getting highest occupancy rate of capacity of a container is very important for the companies, which deals in shipping or has shipping as a part of their main activities. They have to fit 3D boxes in container with optimum or nearest to optimum placement in order to ship more products with a minimum cost. The problem of fitting the boxes which is different from or the same to each other into a big container in optimum level, is called 3-dimensional packing problem. In this problem, the main objective is to minimize used container volume or wasted container space. This provides the reduction of costs in shipments with the use minimum number of containers.

Keywords: Packaging, optimization problems, three-dimensional packing, combinatorial problem, transportation.

1. INTRODUCTION

The best utilization of capacity of a vehicle reduces the number of vehicles used in a shipment. This is very important for companies especially deal in transportation and/or exporting. In order to transport more packages with less number of vehicles, boxes should be packed optimally or near optimally. So, this helps to reduce the transportation costs. The problem of packing all boxes into a container is called 3D packing problem. The 3D packing problem is very complicated. If number of boxes increases, the complexity of the problem increases with nondeterministic polynomial time George and Robinson (1980). Therefore, the packing problem is a type of NP-Hard problem.

For the solution method of 3D packing problem, many different methods are suggested by George and Robinson (1980), NGoi et.al. (1994), Eley (2002) and Pisinger (2002). Data structures methods (Brunetta and Gre'goire (2005)), some meta-heuristic methods like genetic algorithm (Gehring and Borthfeldt (1997)), simulated annealing (Lai and Chan (1996)), some heuristic methods (Fareo at.al. (1999)) and some other efficient algorithms (faina(1999) and Mal et.al. (2005)) is suggested to solve this problem.

3D packing problem is usually solved using optimization and heuristics methods. Some studies assume an unlimited capacity for a container and try to solve 3D packing problem with similar or different sized boxes from each other (Martello et.al. (1997)). Some researchers consider other measures such as shipment of boxes that are more profitable than others boxes. In this case, the more profitable boxes have a priority to be placed into containers (Fareo et.al. (1999)). Some researchers also accept one of dimensions unlimited and the others fixed (Fareo et.al (1999)). In this paper, we assume an unlimited height for a container. The width and depth of the container is computed by considering the size of boxes to be placed.

In this paper, we propose a new heuristic algorithm to solve 3D packing problem. This is organized as follows: Section 2 gives a mathematical model of the problem.

2. PROPOSED ALGORITHM

In this section, we first define the inputs and outputs for the algorithm with our assumptions and then describe the proposed algorithm. The algorithm is called Largest Area First-Fit (LAFF) minimizing height. The algorithm places the boxes with largest surface area first by minimizing height from the bottom of the container.

1.1 Inputs for LAFF Algorithm

The first input is the number of different sized boxes denoted it by N. The second input is the dimensions for each type of different sized boxes. We denote this input parameter with four values (an, bn, cn, kn) where an is width, bn is depth, cn is height and kn is the number of boxes for a given size.

Depending upon how you view a box, each dimension can be considered as width, height or depth. If you assume bn as width, other dimensions an and cn can be accepted as height or depth as well. Because, the box is three dimensional and it can be rotated and be viewed from different perspectives. The following lists the input parameters of the proposed algorithm:

Number of different sized Boxes ..... : N
Width of nth box ..... : an
Depth of nth box ..... : bn
Height of nth box ..... : cn

Number of nth box.....: kn

1.2 Outputs for LAFF Algorithm

After execution of LAFF algorithm, the program produces the following outputs:

O1: The volume of container
O2: The used space (The volume of all boxes placed)
O3: The wasted space
O4: The running time

The output of the running time is expressed in order of milliseconds.

1.3 How the LAFF Algorithm works

As mentioned before, 3D packing problem is combinatorial problem with NP-hard complexity. That means the optimum results for the solution of the problem can be found by trying all combinations of possible different solutions. However, if the number of boxes increases, the number of iterations will increase so much that it cannot be solved in polynomial time on even fastest computer that is available with the current technology. Under some basic assumptions and constraints, these kinds of problems can be solved by heuristic algorithms that provide solution near to optimum.

The proposed algorithm is called Largest Area First-Fit (LAFF) minimizing height. It uses a heuristics that places the boxes with largest surface area first by minimizing height from the bottom of the container. The algorithm works as follows:

First of all, the width and depth of the container must be determined. Given a set of different sized boxes, we compute the width and depth of the container by finding the longest two edges of given boxes. The width and depth of the container is determined at the beginning of the algorithm and it remains fixed throughout execution of the algorithm. We assume the height is unlimited. The height increases as the algorithm runs.

That is, the width (ak) and depth (bk) of the container is determined by selecting the first and second longest edge of given boxes (ai, bi and ci). The longest edge is taken as width (ak) and second longest edge is taken as depth (bk). So, ak and bk never changes.

After determining the values of width and depth of container, the given boxes can be placed into the container. In this algorithm, two types of placement methods are used. The first placement method allocates a space for a box that increases the height of the container. The second placement method allocates space for the remaining boxes if there is a box fits in the available space around the placed box without overflowing the height of the placed box.

In the first placement method, the boxes with the largest surface area are determined and the selected boxes are searched to find a box that has minimum height out of all selected boxes. Then, the box with minimum height is placed in the container. The largest surface of the selected box should be parallel to the bottom of the container (Fig. 1).

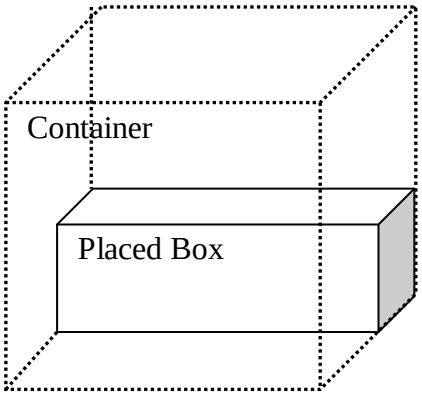

Fig. 1 First type of placement method

In the second placement method, the algorithm tries to allocate space for the remaining boxes around the box that is placed with the first method. That is, If some space remains around the placed box as in Fig. 1, the algorithm tries to fill that space with the second type of placement method. In this method, given that the placed box has dimensions ai, bi and ci on a level, then the container will have two empty spaces around the placed box with dimensions ((ak-ai), bk, ci) and (ak, (bk-bi), ci). So, if there is any box fitting in these empty spaces, that can be one or more, we place the box which has the maximum volume out of fitting boxes as shown in Fig. 2. This iteration continues until there are no boxes or no boxes can fit into those spaces.

In the second placement method, if there is no space around the placed box, then the algorithm continues with the first type of placement method, consequently.

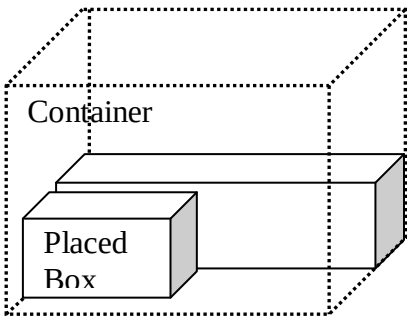

Fig. 2 Second type of placement method.

Whenever there are boxes that are not placed yet, the algorithm goes on with the first type of placement method and so on until all boxes are placed. At the end of the algorithm, a possible solution as shown in Fig. 3 is produced.

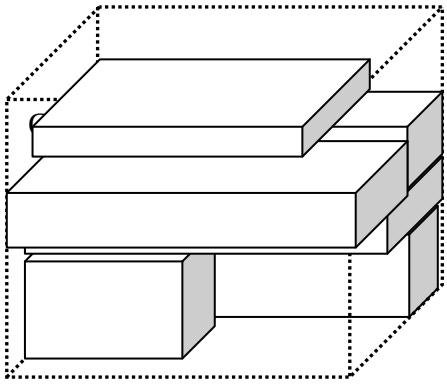

Fig. 3 A possible solution of the Algorithm.

1.4 The LAFF Algorithm

The main steps of the LAFF algorithm are summarized below:

Step1: Input box dimensions and numbers.

N: number of unique boxes.

Step2: Determine the width(ak) and depth(bk) of

The container.

ak : First longest edge of all boxes.

bk : Second longest edge of all boxes.

ck : 0 (zero).

Step3: Chose the box which has the widest surface area. If there is more than one, chose the box which has minimum height. Place this box (ith) on the largest surface parallel to the base of container.

Step3.1: Determine the height of container and decrement the number of ith box.

ck=ck+ ci

ki=ki-1

Step3.2: if the number of Boxes is zero then terminate.

$$\sum_p k_p = 0$$

Step 3.3: if the space (ak-ai)=0 and (bk-bi)=0 then go to Step 3. Otherwise, chose the box which fits into this space. If there is no box that fits into this space, go to Step 3. If there is more than one box that fits into this space, chose the box which has the biggest volume. It called jth box.

Step3.3.1: Determine the dimension of the space.

as = ak–ai –aj ve bs = max(bk–bi, bk–bj)

or

as = max(ak–ai, ak–aj) and bs = bk–bi–bj

Step3.3.2: Decrement number of ith box.

kj=kj-1

Step3.3.3: if the number of Boxes is zero then terminate. Otherwise, go to step 3.3.

$$\sum_p k_p = 0$$

3. COMPLEXITY OF LAFF ALGORITHM

If we denote n as the number of all boxes to be placed into the container and k as types of different sized boxes, the worst-case running time of the algorithm is expressed below:

$T(n) = n(nk)$
$T(n) = kn^2$ where k is constant.

Therefore, the complexity of the algorithm is $O(n^2)$.

4. COMPUTATIONAL EXPERIMENTS

We tested the program with several numbers of boxes with different sizes in order to see practical performance of the algorithm. The program generates random boxes with different sizes as an input and then places the boxes into the container. Table 1 shows the results of experiments. The first column (A) is the number of different types of boxes. The second column (B) is the total number of boxes. The third column (C) is the volume of all placed boxes. The forth column (D) is the volume of constructed container. Finally, the last column (E) shows the percentage amount of the wasted space. The algorithm tries to place boxes in a best way. The amount of the wasted space is considerably low. As it can be seen from the table, when the types of boxes increase, the wasted space will also increase. Because, it is more difficult to place the different sized boxes into the container than that of the same sized boxes.

Table 1. Sample box placement results

Number of Different Boxes (A)	Total Number of Boxes (B)	Volume of placed Boxes (C)	Volume of Container (D)	Waste (%) (E)
1	10	8400	8400	0
1	20	12480	12480	0
2	5	2547	2610	2,41
2	10	6252	6480	3,52
2	15	25554	25920	1,41
2	20	49032	49320	0,58
5	5	3910	5100	23,3
5	10	11359	13680	16,97
5	20	22596	25840	12,55
10	10	13419	19200	30,11
10	20	21694	27740	21,80
10	30	12854	16800	23,49

The program is coded with C programming language. The coded program is run in a computer on with the following configuration: Intel Core Duo 1.73 GHZ CPU 1536 MB Ram. The Program executes very fast. The execution time of all experiments is below 16 milliseconds. We also tested the

program with 10000 boxes with 20 different sizes. In this case, the program execution time is about 380 milliseconds.

5. CONCLUSION

The 3D packing problem is a well-known NP hard problem, which is too complex to be solved exactly in polynomial time. This paper presents a heuristic algorithm for the 3D packing problem. The proposed algorithm uses a heuristics strategy that places the boxes with largest surface area first by minimizing height from the bottom of the container. This algorithm assumes that the height of the container is unlimited and there is no restriction on the orientation of boxes, that is, all boxes can be rotated around any of the three dimensions.

Several conclusions can be drawn from the proposed heuristic algorithm and experiments. The proposed algorithm is an efficient algorithm which is the running time in the order of $O(n^2)$. The amount of wasted space is also in acceptable ranges.

As a future work, other parameters such as weight of boxes, the distribution of the weight in a container, the use of multiple containers, the order of shipment of boxes can be considered to develop new algorithms.

6. REFERENCES

L., Brunetta, P., Gre'goire (2005), "A general purpose algorithm for three-dimensional packing." *INFORMS Journal on Computing*, vol.17, pp. 328-338

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

L., Faina (1999), "A global optimization algorithm for the three-dimensional packing problem", *European Journal of Operational Research*, **126**, 340-354.

O., Fareo, D., Pisinger, M., Zachriasen (1999), Guided Local Search for the Three-dimensional Bin Packing Problem. *Technical Report 99-13*, Dept. of Computer Science, University of Copenhagen, Copenhagen, DK.

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

J. A. George and D.F. Robinson (1980), "A heuristic for packing boxes into a container." *Computer and Operations Research*, **7** (3), 147-156.

K. K. Lai, J.W.M., Chan (1996), "Developing a Simulated Annealing Algorithm for the cutting stock problem." *Computers and Industrial Engineering*, **32** (1), 115-127.

Y. Mal, X., Hong, S., Dong, C.K., Cheng (2005), "3D CBL: An Efficient Algorithm for General 3-Dimensional Packing Problems." Midwest Symp. on Circuits and Systems,pp. 1079-1082

S. Martello, D., Pisinger, D., Vigo (1997), "Three dimensional bin packing problem", *Technical Report DEIS-OR-97-6*, Bologna University.

B. K. A. NGoi, M. L. Tay, and E. S. Chua (1994), "Applying spatial representation techniques to the container packing problem." *International Journal of Production Research*, **32** (1), 111-123.

D. Pisinger (2002), "Heuristics for the container loading problem." *European Journal of Operational Research*, **141** (2), 382-392.