Use of Cartesian Coordinate Geometry in Solving One-Dimensional Bin-Packing Problems

Niluka P. Rodrigo[a*], WB Daundasekera[a], AAI Perera[a]

Department of Mathematics,

Faculty of Science, University of Peradeniya, Sri Lanka.

Abstract

In this paper, our objective is to investigate a mathematical formulation of solving the Bin Packing Problem (BPP). It has become one of the leading mathematical applications throughout the time. A BPP basically describes in three ways; One-Dimensional, Two-Dimensional and Three-Dimensional Bin Packing problems. In our approach, Modified Branch and Bound Algorithm (MBBA) is developed to generate all the feasible packing patterns of given boxes to required containers for One-Dimensional BPP. Further development of algorithms was made to ascertain the locations of each box within the container by using Cartesian coordinate system. Developed algorithms are coded and programmed in the Python programming environment to generate feasible packing patterns.

Keywords: BPP, MBBA, Python Software Package

Language: English

Date of Submission: 2018-03-14

Date of Acceptance: 2018-04-10

Date of Publication: 2018-04-30

Journal: *MATHLAB Journal*

Website: https://purkh.com

1 Introduction

Due to industrial revolution, at present most of the industries are focusing in the frame of globalization. In turn, industries have substantially increased the scope and magnitude of their global production and distribution network around the world. Thus, the globalized market results in rapid development of international trade and creates intensive competitions among the industries. In this research, our objective is to investigate a mathematical formulation of solving the One-Dimensional BPP which has been studied by many researchers in the recent past. Packing the produced goods in the optimum manner within a limited space has multiple bene_ts to the production plant. This will eventually reduce the production of the plant. A Bin Packing problem basically describes in three ways; One-Dimensional, Two-Dimensional and Three-Dimensional Bin Packing problems. The Bin Packing problem can be defined as a finite collection of items with varying specifications to be packed into one or more containers utilizing the maximum volume of the containers while satisfying the supply-demand. Each container can hold any subset of the collection of objects without exceeding its capacity. Bin packing is also called as container loading, box packing, cargo loading, knapsack, etc.

A burning issue faced by the industries is how to find the optimum layout (packing arrangement) of boxes or packing items which have different shapes and sizes within the available bins (container) such that improve the utilization ratio of bins or minimize the bin slack without overlapping the packages. Jatinder N. D. Gupta and Johnny C. (1999) have described a new Heuristic Algorithm to solve the one-dimensional Bin Packing problem. Effectiveness of the proposed algorithm has been compared with First Fit Decreasing (FFD) and Best Fit Decreasing (BFD) algorithms using five different data sets [1]. Mukhacheva E. A et al (2000) developed two algorithms for one-dimensional cutting-packing problem, namely, a modified Branch-and-Bound method (MBB) and Heuristic Sequential Value Correction (SVC) method. Efficiency of the algorithms is discussed from the computational experiment and it seems that the efficiency of the SVC method appears to be superior to that of the MBB [2]. Fleszar K. and Khalil S. presented a new Heuristic Algorithm for minimum bin slack. In this research, each packing is determined in a search procedure that tests all possible subsets of items on the list which _t the bin capacity (C). The slack in packing A is expressed by $S(A) = C$ and updating every time an item is added or removed from A [3]. Andrea Lodi et al (2002) have emphasized on exact algorithms and effective heuristics and metaheuristic approaches to the two-dimensional Bin Packing problem [4].

Christian Blum and Verena Schmid (2013) have dealt with two-dimensional Bin Packing problem under free guillotine cutting, a problem in which a set of oriented rectangular items are given which must be packed into minimum number of bins of equal size. An evolutionary algorithm has been discussed and the results of the proposed algorithm are compared with some of the best approaches from the literature [5]. There is intercommunication in between Cutting Stock Problem (CSP) and BPP. Many researchers have worked on the cutting stock problem as well and designed different algorithmic approaches to solve the problem. Among them, Saad (2001) modified Branch and Bound Algorithm to find feasible cutting patterns for one-dimensional cutting stock problem and formulated a mathematical model to minimize the total cut loss. Modified Branch and Bound Algorithm is illustrated using a case study [6]. Further Rodrigo et al (2012) developed an algorithm, based on Modified Branch and Bound algorithm [6] to determine the feasible cutting patterns for Two-Dimensional cutting stock problem with rectangular shape cutting items. The method was illustrated with the use of a case study, where the data were obtained from a floor tile company known as Mega Marble Company located in London. A computer programme was coded using MATLAB inbuilt functions [7]. As an extension of the above study, Rodrigo et al (2012) redesigned the developed algorithm [7] to determine the locations of each cutting item using Cartesian Coordinate Geometry [8]. Rodrigo et al (2013) modified the Branch and Bound Algorithm to decide the feasible cutting patterns for Two-Dimensional cutting stock problem with triangular shape cutting items and illustrated the algorithm using a case study. The algorithm has been coded using MATLAB environment [9]. Thereafter, Rodrigo et al (2014) has made an approach to nest circular shaped cutting items within rectangular shaped main sheet with known dimensions. Cartesian coordinate points of

centre of each circle have been determined using a computer programme coded in MATLAB software environment [10]. Besides, identification of cutting-packing location within the container (or main sheet) is significantly crucial to outline items within the container depending on the selected cutting-packing pattern. To address this issue, further development of MBBA will be made to ascertain the locations of each cutting-packing item within the main sheet (or container) by using Cartesian coordinate system. Developed algorithm will be coded and programmed in the Python Software environment to generate feasible packing patterns for One-Dimensional BPP.

2 Materials and Method

2.1 One-Dimensional (1D) BPP

In most industries, the cost of the transportation consists of a high percent of total price. The BPP is one of the most famous operation research problems which are defined to improve the amount of packing boxes in to containers. This BPP can be described as follow; there is a container with the rectangular shape base with the width W and the length L. At the same time there are different size of boxes with the rectangular shape base with the width w_i; i = 1, 2, ..., n and the length L. In this paper, MBBA applied to generate feasible packing patterns of n number of boxes to bunch into the container.

2.2 Mathematical Model (Gilmore and Gomary, 1961)

Any firm's main objective is to maximize the annual contribution margin accruing from its production and sales. By reducing costs, wastages and maximizing sales, productivity can be improved. Cost can occur in many ways and transportation cost takes a big challenge in industry. According to the selection, a mathematical model to minimize the packing cost is formulated as follows:

$$\text{Minimize} \quad z = \sum_{j=1}^{n} c_j x_j \quad (\text{Total bin slack})$$

$$\text{Subject to} \quad \sum_{j}^{n} a_{ij} x_{ij} \geq N_j \quad \text{for } i = 1, 2, \ldots, m$$

$$x_{ij} \geq 0 \text{ and Integer for all } i, j.$$

Where

n = Number of boxes with different sizes,

m = Number of patterns,

a_{ij} = The number of occurrences of the i^{th} size box in the j^{th} pattern,

c_j = Bin slack in the j^{th} pattern,

N_j = Number of boxes of the i^{th} size box,

x_{ij} = The number of boxes of i^{th} size box can be packed in j^{th} pattern.

2.3 Modified Branch and Bound Algorithm (MBBA)

Step 1: Arrange required widths (or lengths) of boxes, w_j , j = 1, 2, . . . ,n in decreasing

order, ie w_1 , w_2 , ..., w_n , where n = number of boxes from each sizes.

Step 2: For i = 1, 2,. . . ,m and j = 1 do Steps 3 to 5.

Step 3: Set

$$a_{11} = \left[\left[W / w_1 \right]\right]; \quad \rightarrow (1)$$

$$a_{ij} = \left[\left[\left(W - \sum_{z=1}^{i-1} a_{zj}\, w_z \right) / w_i \right]\right]; \quad \rightarrow (2)$$

where W is the width of the container. Here, a_{ij} is the number of quantity of the i^{th} box in the j^{th} pattern along the width of the container and [[y]] is the greatest integer less than or equal to y.

Step 4: Set $P_j = a_{ij}$

where P_j is the number of quantities of the i^{th} box in the j^{th} pattern within the base of the container.

Location of the i^{th} item in the j^{th} pattern and number of pieces from each item:

If $a_{ij} > 0$, then set

$$(x_i, y_i) = \left(\left(\sum_{z=1}^{i} a_{zj}\, w_z - a_{ij}\, w_i \right), 0 \right); \left(\left(\sum_{z=1}^{i} a_{zj}\, w_z \right), 0 \right)$$

$$N_i = a_{ij}$$

where N_i is the number of pieces can be nest from the i^{th} box in the j^{th} pattern along the width-wise and (x_i, y_i) , are the coordinates of the each box of i^{th} item in the j^{th} pattern within the container.

Step 5: Bin slack along the width of the container:

$$C_j = \left(W - \sum_{i=1}^{m} a_{ij}\, w_i \right); \quad \rightarrow (3)$$

Step 6: Set $t = m$-1.

While t $\geq$ 0 , do Step 7.

Step 7: While $a_{tj} \geq 0$ set $j = j + 1$ and do Step 8.

Step 8: Generate a new pattern according to the following conditions:

For z = 1, 2, ..., t – 1;

Set $a_{z\,j} = a_{z(j-1)}$

For z = t

Set $a_{z\,j} = \left(a_{z(j-1)} - 1\right)$

For z = t + 1, ..., m

calculate a_{zj} using (1) .

Go to Step 4.

Step 10: Set t = t-1.

Step 11: STOP.

2.4 Case Study

Proposed MBBA to solve one-dimensional BPP is tested and analyzed to determine feasible and optimal packing patterns. Following examples will illustrate how to generate feasible packing patterns by minimizing the total bin slack.

Following data table represents five one-dimensional bin packing problem given in the paper [1] and proposed algorithm in this paper has been applied to solve those problems.

Problem No	Width of the boxes	Width of the base of the container
1	60, 20, 50, 30	100
2	3,2	7
3	7,4,5,3	13
4	9,17,6,5,4,7	17
5	24, 22, 44, 21, 8, 17, 6	61

Table 1: Required widths of the boxes and containers

3 Results and Discussion

Modified Branch and Bound Algorithm is applied to the above examples to generate feasible packing patterns as given below:

Consider the Problem No 1 in the Table 1.

Item No	1	2	3	4
Required widths	60	20	50	30
Demand	1	3	1	1

Table 2: Required widths and demand of problem 1.

Following table represent the generated feasible packing patterns applying MBBA for problem 1 given in the table 2. Python programming language is used to code the MBBA.

Pattern No / Width of the boxes	1	2	3	4	5	6	7	8	9
60	1	1	0	0	0	0	0	0	0
50	0	0	2	1	1	0	0	0	0
30	1	0	0	1	0	3	2	1	0
20	0	2	0	1	2	0	2	3	5
Bin Slack	10	0	0	0	10	10	0	10	0

Table 3: Generated Packing Patterns of problem 1

====== RESTART: C:\Users\NILUKA\Documents\PhD\python programmes\One.py ===

Width of the base of the Container = 100

insert how many box you want:4

Enter width of the boxes in decending order:60

Enter width of the boxes in decending order:50

Enter width of the boxes in decending order:30

Enter width of the boxes in decending order:20

[60, 50, 30, 20]

Pattern 1 = [1, 0, 1, 0]

Number of boxes can be packed from 1 size box = 1

Coordinates =(0 , 0);(60 , 0)

Number of boxes can be packed from 2 size box = 0

Number of boxes can be packed from 3 size box = 1

Coordinates =(60 , 0);(90 , 0)

Number of boxes can be packed from 4 size box = 0

Bin slack for pattern 1 = 10

Pattern 2 = [1, 0, 0, 2]

Number of boxes can be packed from 1 size box = 1

Coordinates =(0 , 0);(60 , 0)

Number of boxes can be packed from 2 size box = 0

Number of boxes can be packed from 3 size box = 0

Number of boxes can be packed from 4 size box = 2

Coordinates =(60 , 0);(100 , 0)

Bin slack for pattern 2 = 0

Pattern 3 = [0, 2, 0, 0]

Number of boxes can be packed from 1 size box = 0

Number of boxes can be packed from 2 size box = 2

Coordinates =(0 , 0);(100 , 0)

Number of boxes can be packed from 3 size box = 0

Number of boxes can be packed from 4 size box = 0

Bin slack for pattern 3 = 0

Pattern 4 = [0, 1, 1, 1]

Number of boxes can be packed from 1 size box = 0

Number of boxes can be packed from 2 size box = 1

Coordinates =(0 , 0);(50 , 0)

Number of boxes can be packed from 3 size box = 1

Coordinates =(50 , 0);(80 , 0)

Number of boxes can be packed from 4 size box = 1

Coordinates =(80 , 0);(100 , 0)

Bin slack for pattern 4 = 0

Pattern 5 = [0, 1, 0, 2]

Number of boxes can be packed from 1 size box = 0

Number of boxes can be packed from 2 size box = 1

Coordinates =(0 , 0);(50 , 0)

Number of boxes can be packed from 3 size box = 0

Number of boxes can be packed from 4 size box = 2

Coordinates =(50 , 0);(90 , 0)

Bin slack for pattern 5 = 10

Pattern 6 = [0, 0, 3, 0]

Number of boxes can be packed from 1 size box = 0

Number of boxes can be packed from 2 size box = 0

Number of boxes can be packed from 3 size box = 3

Coordinates =(0 , 0);(90 , 0)

Number of boxes can be packed from 4 size box = 0

Bin slack for pattern 6 = 10

Pattern 7 = [0, 0, 2, 2]

Number of boxes can be packed from 1 size box = 0

Number of boxes can be packed from 2 size box = 0

Number of boxes can be packed from 3 size box = 2

Coordinates =(0 , 0);(60 , 0)

Number of boxes can be packed from 4 size box = 2

Coordinates =(60 , 0);(100 , 0)

Bin slack for pattern 7 = 0

Pattern 8 = [0, 0, 1, 3]

Number of boxes can be packed from 1 size box = 0

Number of boxes can be packed from 2 size box = 0

Number of boxes can be packed from 3 size box = 1

Coordinates =(0 , 0);(30 , 0)

Number of boxes can be packed from 4 size box = 3

Coordinates =(30 , 0);(90 , 0)

Bin slack for pattern 8 = 10

Pattern 9 = [0, 0, 0, 5]

Number of boxes can be packed from 1 size box = 0

Number of boxes can be packed from 2 size box = 0

Number of boxes can be packed from 3 size box = 0

Number of boxes can be packed from 4 size box = 5

Coordinates =(0 , 0);(100 , 0)

Bin slack for pattern 9 = 0

Number of patterns = 9

The mathematical model is solved as an integer programming model using EXCEL to design generated packing patterns so that the bin slack will be minimized and optimum solution to the model is given in the following table.

Optimum Pattern No	2	4	Demand
Width of the boxes			
60	1	0	1
50	0	1	1
30	0	1	1
20	2	1	3
No of Containers	1	1	

Table 4: Optimum solution for problem no 1

Problem No 2 in Table 1:

There are 3 different packing patterns according to the MBBA and two containers should be required to satisfy the demand of boxes.

Optimum Pattern No	2	Demand
Width of the boxes		
3	1	2
2	2	4
No of Containers	2	

Table 5: Optimum solution for problem no 2

Problem No 3 in Table 1:

There are 11 different packing patterns according to the MBBA and three containers should be required to satisfy the demand of boxes.

Optimum Pattern No	3	5	10	Demand
Width of the boxes				
7	1	0	0	1
5	0	1	0	1
4	0	2	1	3
3	2	0	3	5
No of Containers	1	1	1	

Table 6: Optimum solution for problem no 3

Problem No 4 in Table 1:

There are 18 different packing patterns according to the MBBA and five containers should be required to satisfy the demand of boxes.

Optimum Pattern No / **Width of the boxes**	**1**	**5**	**7**	**17**	**Demand**
17	1	0	0	0	1
9	0	1	0	0	1
7	0	0	1	0	1
6	0	0	1	0	1
5	0	0	0	1	2
4	0	2	1	3	9
No of Containers	1	1	1	2	

Table 7: Optimum solution for problem no 4

Problem No 5 in Table 1:

There are 71 different packing patterns according to the MBBA and three containers should be required to satisfy the demand of boxes.

Optimum Pattern No / Width of the boxes	1	5	7	Demand
44	1	0	0	1
24	0	1	1	2
22	0	1	0	1
21	0	0	1	1
17	1	0	0	1
8	0	0	2	2
6	0	2	0	2
No of Containers	1	1	1	

Table 8: Optimum solution for problem no 5

Effectiveness of the proposed MBBA has been compared with algorithms given in the paper [1].

No of Containers from different algorithms / Problem No	MBBA	MBS	FFD	BFD
1	2	2	3	3
2	2	2	3	3
3	3	3	4	4
4	5	5	6	6
5	3	3	4	4

Table 9: No of containers for problems given in the paper [1]

4 Conclusion

In this paper, a bin packing problem is formulated as a mathematical model based on the concept of packing patterns. Five different problems are solved using proposed MBBA and results (minimum number of containers to satisfy the demand of each box) are compared with three different algorithms; Minimum Bin

slack (MBS), First Fit Decreasing (FFD) and Best Fit Decreasing (BFD). According to the MBS algorithm [1], Minimum containers are found to satisfy the demand of each size of boxes. In our approach; MBBA gives comparable solution with MBS. However, looking the solution of the MBS, it's difficult to identify the places of each box should be allocated to into container satisfying the minimum number of containers for each problem. As a result, identification of packing location within the base of the container is significantly vital to trace boxes within the container depending on the selected packing pattern. Accordingly, MBBA can be addressing this issue; to ascertain the locations of each packing item within container by using Cartesian coordinate system. Developed algorithm (MBBA) is coded and programmed in the Python programming environment to generate feasible packing patterns and location of each box in the pattern.

References

[1] Jatinder N. D Gupta and Johnny C. "*A new heuristic algorithm for the one-dimensional bin-packing problem*", Production planning and Control, ISSN 0953-7287 (1999),Vol. 10, No. 6, 598-603.

[2] Mukhacheva E. A, G. N Belov, V.M. Kartack and Mukhacheva A.S. "*Linear one-dimensional cutting-packing problems: Numerical experiments with the sequential value, Correction Method and a modified Branch and Bound method*",

[3] Fleszar K. and Khalil S. "*New Heuristics for one-dimensional bin packing*", Research-Gate: Computers and Operations Research, DOI: 10.1016/S0305-0548(00)00082-4.

[4] Andrea L, Silvano M and Daniele V. "*Recent Advances on two-dimensional bin packing Problems*", Elsevier Science, Discrete Applied Mathematics. 123 (2002), 379-396.

[5] Christian B. and Verena S. "*Solving the 2D bin packing problem by means of a Hybrid Evolutionary Algorithm*", Elsevier, International Conference on Computational Science, ICCS (2013), 899-908.

[6] Saad M. A. Suliman. "*Pattern generating procedure for the cutting stock problem*", International Journal of Production Economics 74 (2001) 293-301.

[7] W. N. P. Rodrigo, W. B. Daundasekara and A. A. I. Perera. "*Pattern Generation for Two- Dimensional Cutting Stock Problem*", International Journal of Mathematics Trends and Technology, Vol.3, Issue 2: 54-62.

[8] W. N. P. Rodrigo, W. B. Daundasekara and A. A. I. Perera. "*Pattern Generation for Two-Dimensional Cutting Stock Problem with Location*", Indian Journal of Computer Science and Engineering (IJCSE), Vol. 3, No 2, April-May 2012, 354-368 (www.academia.edu/4463690/INDJCSE12-03-02-082).

[9] W. N. P. Rodrigo, W. B. Daundasekara and A. A. I. Perera. "*A Method for Two- Dimensional Cutting Stock Problem with Triangular Shape Items*", British Journal of Mathematics and Computer Science (BJMCS), 3(4): 750-771.

[10] W. N. P. Rodrigo, W. B. Daundasekara and A. A. I. Perera. "*Optimum Arrangement in Two-Dimensional Cutting Stock of Circular Items*", Postgraduate Institute of Science Research Congress, Sri Lanka