A hybrid multi-objective genetic algorithm for the container loading problem

Nivedha Ramesh, Jagadeeshwaran Raja Umashankar

Abstract—Three dimensional bin packing problems (3D-BPP) arise in industrial applications like container ship loading, pallet loading, plane cargo management, warehouse management, etc. In this paper we propose to use a hybrid genetic algorithm to solve a single container loading problem, by optimizing the utilized volume, the number, and total value of the contained boxes. Within the framework of the proposed algorithm, a special diploid representation scheme of individual is used and a modified heuristic packing method, derived from the deepest bottom left with fill (DBLF) packing method, is employed to realize a 3D packing of the boxes. An experimental study over a set of self-generated 3D-BPP test problems shows that the proposed algorithm is efficient and adaptable to address 3D-BPP.

Index Terms—Container packing, 3D-BPP, elitism, genetic algorithms, multi-objective optimization, NSGA II, Pareto-optimal solutions

1 Introduction

The bin packing problem (BPP) can be defined as finding an arrangement for n objects, each with some weight, into a minimum number of larger containing objects (or bins), such that the total weight of the item(s) in the bin does not exceed the capacity of the bin. There are many variants of the BPP, of which the three-dimensional bin packing problem (3D-BPP) is the most challenging and important. 3D-BPP directly models several issues in production and transportation planning and has many applications in container ship loading, cargo management, pallet loading, and warehouse management.

Inspired from natural evolution, genetic algorithms (GA) have been consistently deployed to solve many real-world optimization problems. However, the increasing complexity of such problems considerably reduces the performance of these algorithms, giving rise to the hybridization of a GA and a problem-dependent heuristic method. Such hybrid GA(s) have proved to be more efficient [1,2].

In this paper we focus on solving the Container Loading Problem (CLP) using a single container with a strong emphasis on weakly and strongly heterogeneous set of boxes. More precisely, given the dimensions of a container and a set of boxes with a specific value associated with each box, we try to find an optimal arrangement of these boxes so as to optimize the utilization volume of the container, the number of boxes and the total value of the boxes being packed onto the container. In real-world applications such as supply chains, CLP involves optimizing many such objectives and constraints. Loading these containers efficiently (minimizing the empty spaces inside them), is an economic necessity.

In our algorithm, we adopt a diploid representation of chromosomes, to account for the order and the rotation value of the boxes. A modified variant of the deepest bottom left with fill packing algorithm is designed to convert the individuals into a 3D solution and evaluate them. We apply NSGA II to this multi-objective problem to generate a set of non dominant solutions. The hybrid genetic algorithm has been extensively tested for variation of the performance with the mutation rates and the average fitness values for different categories of rotation(no rotation, 2-way rotation and 6-way rotation) have been contrasted.

2 Related Work

Several researchers have studied on different versions of the three-dimensional bin packing problem, which focus on packing rectangular items into a minimal number of rectangular containers of identical size. During last decades, both exact and heuristic algorithms for this problem have been developed. While exact algorithm can find optimal solution, it usually needs huge amount of time to solve even just moderate size instances. Heuristic algorithm ,which cannot guarantee optimal, is capable to give good solutions with much less computational effort.

Variations of the 3D-BPP have been studied in literature. The single container loading problem (SCLP) deals with packing a selected subset of items into single bin pursuing high utilization ratio. Kang et al. [11] present a hybrid genetic algorithm for a three-dimensional bin packing problem in which boxes are packed into a single bin to maximize the number of boxes packed.

Optimization of multiple objectives requires that the relative importance of each objective be specified in advance which requires a prior knowledge of the possible solutions. But by using the concept of Pareto-dominance it is possible to avoid the need to know the possible solutions in advance [9]. Knowles and Corne [19] introduced the PAES wherein the parent and offspring as well as the archived best solutions thus far are compared using pareto-dominance. But the NSGA which was first proposed by Srinivas and Deb [20] in 1995 proved to be a landmark in the history of MOEAs. The time-complexity of the simple SGA is O(MN3) where M is the number of objectives and N is the size of the dataset. Soon after, Deb and his students [21] developed a faster variant of SGA, called NSGA-II, whose time-complexity is O(MN2). During the last few years several researchers have come up with variants of NSGA-II which have a time complexity of O(MN logM-1N). The works by Fang [22] and Tran [23] are worth mentioning. Kumar et al. [18] have developed a memetic version of the NSGA-II.

3 Problem Statement

Let us consider the model of shipping packages from a distribution center to different customers. The packages need to be loaded onto a container and then transported to the customers world-wide. The packing problem that we deal with here is a geometrical assignment problem which involves small three-dimensional items, *boxes*, to be packed onto a large three-dimensional object, *container*, such that the objective function is optimized.

Apart from the obvious constraints that the boxes must not overlap while packing and that they must not exceed the dimensions of the container, there are other necessary constraints to adhere to. Such as -

- *Support:* All the boxes must either be placed on the container floor or on top of other boxes. No box can be placed in mid-air
- *Orientation:* Depending on their content, the boxes are of a fixed orientation or can be rotated. The several type of orientations are:
  - Only one orientation is permitted and hence the box cannot be rotated
  - Only vertical orientation is permitted. Rotations on the horizontal plane are permitted
  - Free rotation

3.1 Problem Formulation

In this problem, a container of length L, width W and height H is to be filled with a set of boxes of different types, $i = 1,2,...,n$. It is assumed that the container lies in the first octant of a three-dimensional co-ordinate system with it's bottom left co-ordinate overlapping with the origin. The length, width and height of the container are aligned with the x-, y-, and z-axis, respectively as shown in Fig 1. Each box has a length l_i, width w_i and height h_i. Every box b_i also has a rotational value r_i, depending on the allowed orientations.

The objective here is to find an orthogonal packing of these boxes so as to maximize the number of boxes packed, maximize the volume utilization of the container and maximize the total value of the boxes packed in the container.

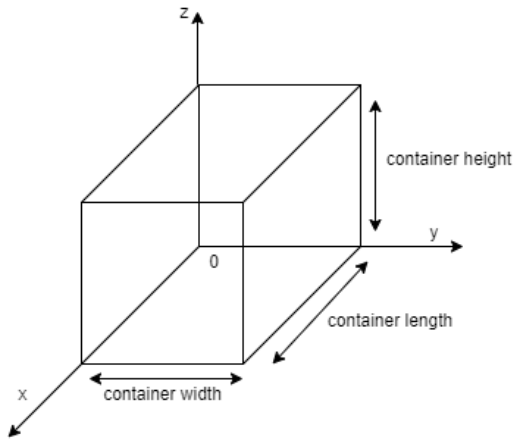

Fig. 1. Placement of a container

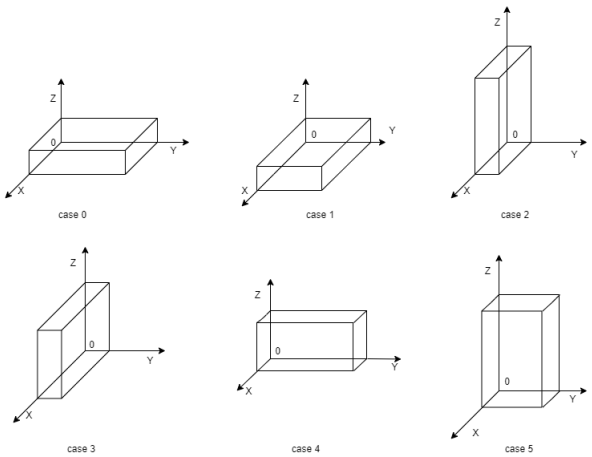

Fig. 2. Six ways of box rotations

4 Proposed Algorithm

A hybrid GA is proposed to solve this multi-objective CLP.

4.1 Representation

Representation of an individual in the genotype space is a crucial factor that affects the performance of a GA. Straight representations like the 0-1 encoding in a knapsack problem, order encoding in travelling salesman problem, have made the translation between the solution and the individual representation easier. However, such representations cannot be used in complex optimization problems.

For example, in our problem, the solution refers to whether and how the boxes will be packed in a container to optimize the utilization volume and the other objectives. Hence we represent every individual by a special diploid chromosome and then use a heuristic packing algorithm to realize the 3D solution. The diploid scheme uses two chromosomes, to represent two different characteristics of the individual. As shown in Fig. 3 chromosome 1 is a simple permutation of the n boxes, denoting the order in which the boxes are to be packed onto the container. Chromosome 2 contains the rotation values for the n boxes according to the allowed rotation category.

x_1 x_2 ..... x_n Chromosome 1
r_1 r_2 ..... r_n Chromosome 2

Fig. 3. Diploid Representation of Individual

In our work, we consider three different rotation categories – no rotation, 2way rotation, and 6way rotation. For boxes that cannot be rotated while being packed into the container $r_i = 0$ *(i=1,2,...,n)*. For boxes that need to be placed upright in the container, $r_i = 0$ or 1 *(i=1,2,....,n)*. Hence if the initial dimension is (l_i, w_i, h_i), the deposited dimension becomes (l_i, w_i, h_i) for $r_i = 0$ and (w_i, l_i, h_i) for $r_i = 1$. For a box that can be rotated without any restrictions $r_i = 0, 1, 2, 3, 4, 5$ and the respective deposited dimensions are (l_i, w_i, h_i), (w_i, l_i, h_i), (l_i, h_i, w_i), (w_i, h_i, l_i), (h_i, w_i, l_i), (h_i, l_i, w_i).

4.2 Packing Algorithm

3D-BPP heuristics can broadly be classified as *Construction* and *Local search* based heuristics. In construction heuristics, the boxes, pre-sorted along any of the dimensions, are placed one by one to an existing partial packing. Generally, researchers use some form of layering techniques like wall building, stack building or block building to pack a 3D bin. But the performance of such algorithms can degrade if the input causes a fragmentation of the surface in the initial stages. In this paper, we use a construction heuristic called the *Simple Deepest Bottom Left with Fill*(S-DBLF) to translate an individual into a three-dimensional solution.

S-DBLF is an extension of the *Bottom Left with Fill* (BLF) proposed by Karabulut and Mustafa in [12]. Every box b in a set of boxes B is moved to the deepest available position

(smallest l value), then as much to the bottom as possible *(smallest h value)* and eventually as far to the left as possible *(smallest w value)*. The pseudo code for the algorithm is shown in Fig. 4.

```
begin
    initialize the position set P;
    for b := 1 to |B| do
    packed := false;
    sort the position set according to the DBL order;
    for j := 1 to |P| do
        if box b € B can be packed into position j € P then
            placed := true;
            set position j as the packed position of box b;
            update the position set P;
            break;
        end if
    end for
    end for
end;
```

Fig. 4. Pseudo code for S-DBLF Algorithm

Initially, a position set P of the container is defined and is set to $(0, 0, 0)$ in the beginning. For every box b in the list B, P is sorted according to the DBL order. The position list is then iterated to check if b can be placed in any of the positions available in P. When a suitable position j is found, set the position j as the packed position of the box b. Placing the box in a rectangular container, gives rise to three distinct cuboid spaces that can be called as the *top-space*, *side-space* and the *front-space* as indicated in Fig. 5. Generally, the algorithm includes two checks: the box b should not exceed the dimensions of the position j and it should not overlap with any of the previously placed boxes [11]. But, the three cuboidal spaces calculated in our algorithm do not overlap, thereby avoiding any additional time consuming computations and introducing a sense of order in the packing. The position set P is then updated with the new available positions. The algorithm stops when all the boxes in the list B have been checked.

Although this method packs the boxes more effectively, it is computationally expensive (O(n3)). Also, placement decisions are made from the local criteria, which may result in a poor packing. But this can be avoided by using a genetic algorithm that maintains population of different packing orders.

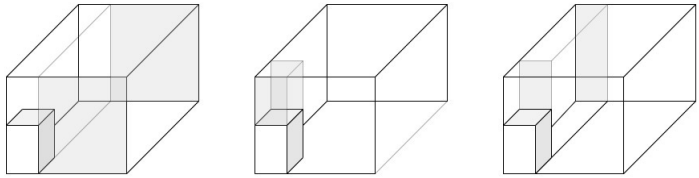

Fig. 5. Cuboid spaces generated after inserting a box

4.3 Crossover and Mutation

As order based crossover has proved to be suited for a permutation encoding, we have used a modified variant [18] to suit our diploid chromosome representation. Using this method, two child individuals *C1* and *C2* are produced from two parent individuals *P1* and *P2*. We start by selecting two random positions i and j, and the portion of the chromosome in *P1(i)* to *P1(j)* is copied onto *C1(i)* to *C1(j)* while *P2(i)* to *P2(j)* is copied onto *C2(i)* to *C2(j)*. Then *C1* is swept circularly from *j+1* filling the remaining values from *P2* and similarly *C2* is filled with values from *P1*.

P1	3	6	1	0	2	4	5
	2	3	0	0	3	1	0
P2	4	0	2	1	5	3	6
	4	1	3	5	1	0	2
C1	5	3	6	0	2	4	1
	1	0	2	0	3	1	5
C2	0	2	4	1	5	3	6
	0	3	1	5	1	0	3

Fig. 6. Example of modified order crossover for 2 way rotation

The individuals produced by recombination undergo a two step mutation. We perform a 2-OPT mutation with a probability of p_{m1} and then do a random resetting mutation with a probability of p_{m2}. As indicated in Fig. 7, two random positions i and j are selected and the portion of the individual *P(i)...P(j)* is reversed. Further, each bit in the second chromosome is selected with a probability of p_{m2} and is

randomly reset to a different rotation value based on the rotation category.

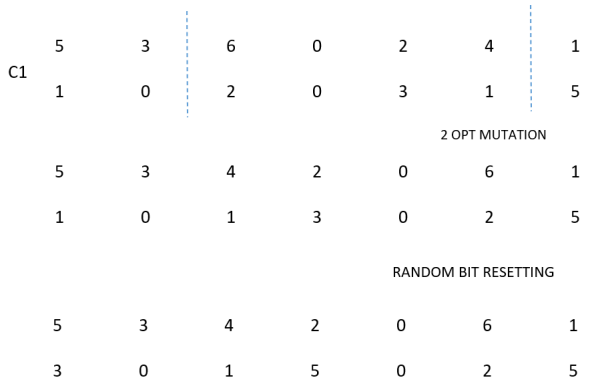

Fig. 7. Example of two step mutation for 2 way rotation

4.4 Multi-objective Optimization

Real-world optimization problems are often multi-objective, making it impossible to arrive at a single best solution. This gives rise to a set of optimal solutions(known as Pareto-optimal solutions), where in the absence of any further information, one cannot tell which is better than the others. Among the existing evolutionary algorithms for solving a Multi-objective optimization problem (MOOP) the most prominent ones are the Pareto archived evolutionary strategy (PAES), Strength pareto evolutionary algorithm (SPEA-2) and the Non-dominated sorting genetic algorithm (NSGA-II)[15]. All these algorithms are based on the concept of Pareto-dominance. NSGA-II has evolved over the last few years with many new variants[16] which have attempted to reduce its time-complexity or improve its convergence to the true Pareto front.

Every individual in the problem context represents a certain order and orientation in which the boxes are to be packed in the container. Each box is also associated with a unique value irrespective of its dimensions. Based on this information we know exactly which boxes have been packed into the container thereby all optimization objectives - volume utilized, number of boxes placed and total value of boxes placed can be evaluated. We use the NSGA-II algorithm to optimize these objectives as it is more efficient in finding a diverse set of solutions and in converging near the true Pareto-optimal set.

The initial population of size M undergoes crossover and mutation to generate child population of size N. Next, we combine the two populations, to form an intermediate population of size $(M+N)$ and the fitness of the individuals is evaluated by the packing algorithm. After calculating the multiple objective fitness values, we carry out non-dominated sorting procedure over the $(M+N)$ population, to rank and divide the individuals into different non-dominated fronts. Thereafter, we create the new parent population by choosing individuals of the non-dominated fronts, one at a time. We choose the individuals of best ranked fronts first followed by the next-best and so on, till we obtain M individuals. In case there is space only for a part of a front in the new population, we use a crowded-distance operator to determine the individuals among those in the front that are from the least crowded regions. We choose such individuals so as to fill up the required number in the new population. Details of crowed-distance operator can be obtained from [15].

4.5 Population Initialization

The initial population is often generated randomly in any standard GA. But to improvise the quality, certain special individuals are generated according to the heuristic that a bigger box should be packed into the container more early. Accordingly, five different individuals based on five different sequences are generated by sorting the values of volume, length, width, height and value of the boxes in descending order. The rest of the individuals are generated in a random order.

5 Experimental Study

5.1 Generation of Test Data

Due to the lack of availability of a standard data-set with volume and value maximization objectives, we used a self generated test data to test the performance of our algorithm. The test data contains *25* problems created using a random heuristic method

TABLE 1
Variation of average fitness with mutation parameters

p_{m1}	p_{m2}	Avg Vol	Avg no	Avg value
0.1	0.01	51.95	73.37	73.34
0.2	0.02	54.11	73.87	74.70
0.3	0.03	52.19	72.17	72.53
0.35	0.04	50.17	71.85	72.77
0.40	0.05	50.37	72.81	73.61
0.45	0.06	51.68	70.24	70.82
0.50	0.10	50.18	69.35	69.70

as indicated in Fig. 8. The generated boxes are a fair mix of weakly and strongly heterogeneous boxes. Every box is assigned a non-zero value v irrespective of its dimensions and position. The data-set used can be downloaded from the url: https://github.com/Nivedha-Ramesh/Container-Loading-Problem

randomly generate *(number of boxes, dimensions of the container)*
where i is the number of loops required
repeat
select randomly *one of the existing boxes*
if start
use object as large box
place random point *on one of the edges*
drop a perpendicular *to the immediate edges*
construct a plane *that intersects the box's space*
create *2 new boxes*
until number of boxes reached

Fig. 8. Algorithm to generate test-data

5.2 Experimental Design

In this paper, the behaviour of the algorithm is examined on a series of test problems, generated using the algorithm shown in Fig. 8. 25 problem sets of weakly and strongly heterogeneous boxes are used to evaluate the performance of the algorithm, under all 3 cases of rotation.

In the proposed GA the initial population of *pop_size* is created with several individuals being created in a heuristic manner and the rest generated randomly. Every individual is converted to a 3D-BPP solution using the S-DBLF algorithm to evaluate the fitness values. The solutions are then ranked using the NSGA II algorithm and a binary tournament selection is performed to select individuals to undergo recombination. The selected individuals undergo order crossover with a crossover probability p_c and the newly generated individuals undergo a two-step mutation. The parent population and the generated population is combined together and then ranked according to the NSGA II algorithm prior to selecting *pop_size* individuals again.

The following parameters are used: the population size (*pop_size*) is set to 36, the modified crossover probability p_c is set to 0.8, the probabilities p_{m1} and p_{m2} in the two-step mutation are set to 0.2 and 0.02 respectively. For each experiment of an algorithm on a test case, 10 independent runs were executed with the same set of random seeds. For each run of an algorithm on a problem, the maximum allowed number of generation is set to 400 and the achieved average fitness is recorded. The overall performance of an algorithm is defined as the achieved best fitness averaged across the number of total runs.

5.3 Experimental Results and Analysis

Upon experimenting with the values of p_{m1} and p_{m2}, it was found that the average fitness of all the 25 problem sets was optimal for the values p_{m1} = 0.2 and p_{m2} = 0.02 as shown in Table 1.

We chose Problem set 10 to be run for 10 times for 400 generations and computed the average values of the fitness over the 10 runs. The utilization percentage of the container is 78.89%, the average number of boxes placed is 83.33% and the average value of the boxes packed is 85.53%, as indicated below.

Problem Set	10
Number of Boxes	19
Container Dimension	262, 71, 258
Avg. Volume	78.892
Avg. Number	83.3325
Avg. Value	85.5324

The average fitness values are plotted over the generations as shown in Fig. 9. The true packing solution for problem set 10 is visualized in Fig. 10. and one of the Rank 1 solutions after 400 generations is visualized in Fig. 11.

TABLE 2
Fitness values for different rotations for pm_1: 0.2 and pm_2: 0.02

Problem Set	No of Boxes	Container Dimension	6-way rotation			2-way rotation			no rotation		
			avg volume	avg no. of boxes	avg value	avg volume	avg no. of boxes	avg value	avg volume	avg no. of boxes	avg value
1	9	215, 291, 519	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
2	33	215, 291, 519	76.03	82.16	84.35	59.76	88.22	87.11	62.16	89.06	88.75
3	28	587, 417, 96	57.59	79.76	75.80	52.29	69.64	64.59	53.90	85.21	79.40
4	18	587, 417, 96	59.58	85.95	88.14	66.56	81.02	83.06	64.67	85.18	86.26
5	14	68, 108, 522	81.03	84.69	84.75	56.38	91.23	91.49	77.59	85.51	84.24
6	17	68, 108, 522	59.59	91.23	91.49	70.40	60.08	69.48	61.58	69.08	75.72
7	22	552, 68, 517	76.36	84.61	84.38	73.94	82.73	85.86	73.50	83.84	81.82
8	10	484, 183, 563	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
9	20	484, 183, 563	76.58	81.09	78.64	69.01	88.33	89.69	73.87	80.21	79.21
10	19	262, 71, 258	76.88	81.58	86.60	46.46	61.40	63.95	53.48	68.42	70.41
11	25	262, 71, 258	51.44	79.00	80.41	46.64	72.50	72.53	51.91	69.39	69.70
12	26	491, 320, 245	73.69	83.46	86.57	59.08	80.93	84.99	72.40	81.22	82.04
13	11	491, 320, 245	60.87	72.73	69.67	59.00	70.21	70.51	60.91	71.47	68.90
14	20	332, 194, 90	74.68	86.61	87.38	68.76	87.29	86.60	64.49	79.43	82.72
15	19	218, 489, 404	73.88	79.44	82.21	69.45	83.04	85.77	59.38	83.22	86.20
16	17	218, 489, 404	62.45	85.76	87.33	76.50	72.94	77.32	78.42	89.00	88.56
17	32	415, 331, 192	63.78	74.19	71.84	50.04	72.92	70.94	61.76	74.22	72.51
18	25	415, 331, 192	67.98	83.06	82.07	58.78	72.44	73.90	67.84	79.44	79.18
19	35	337, 220, 140	69.11	88.39	88.80	67.62	82.03	82.92	75.33	82.38	81.55
20	21	337, 220, 140	74.18	86.09	89.53	75.91	82.31	82.96	55.58	78.57	80.30
21	14	151, 552, 481	85.39	91.27	90.54	83.29	92.86	92.80	84.10	92.66	93.31
22	28	151, 552, 481	72.80	75.23	72.86	66.12	72.56	72.68	73.47	75.71	74.47
23	23	554, 377, 480	76.78	85.33	86.76	63.88	85.63	87.04	72.63	93.72	94.56
24	36	554, 377, 480	64.58	81.42	82.87	62.56	81.82	82.49	60.79	88.09	89.80
25	34	554, 377, 480	68.85	80.14	78.97	61.63	78.67	77.81	59.47	78.76	76.65

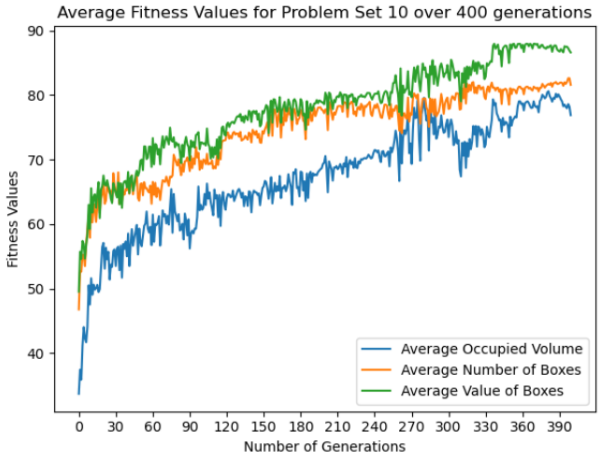

Fig. 9. Fitness variation for Problem set 10

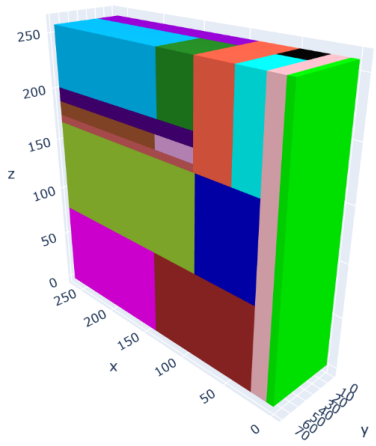

Fig. 10. Visualization of the ideal solution for Problem set 10

The results of the algorithm run for 25 test problems is presented in Table 2, where the average occupied volume(%), average number of boxes packed(%) and the average total value of the boxes packed(%) is shown for the all three categories of rotation. When the number of boxes is lesser than 10, the algorithm successfully packs all the boxes.

For most of the test problems, 6 way rotation provides better results than 2 way or no

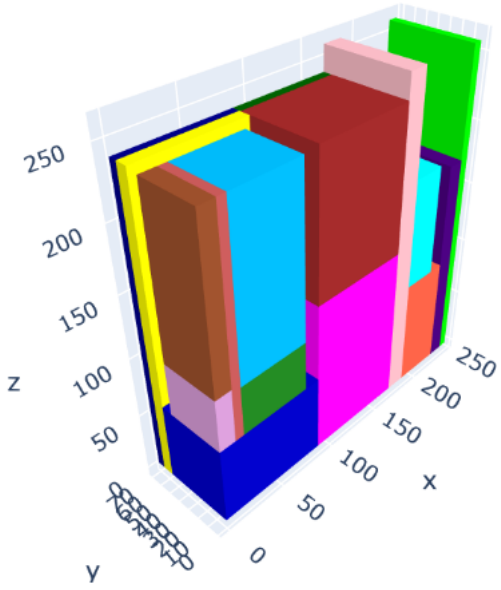

Fig. 11. Visualization of a non-dominant solution after 400 generations for Problem set 10

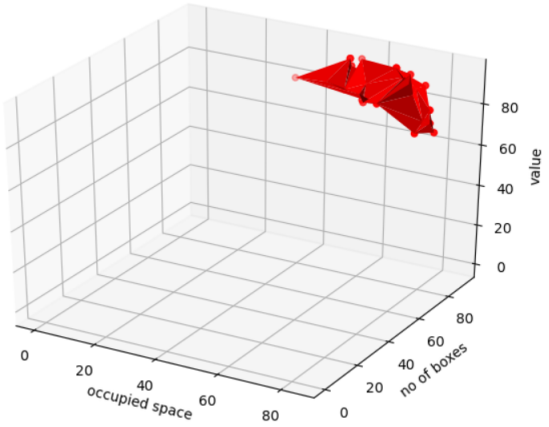

Fig. 12. Pareto front of Problem set 10 after 400 generations

rotation. However, in a few cases the NSGA II algorithm tries to optimize the value and/or number of the boxes packed over the volume of the container occupied. Hence, the results are better for the other cases of rotation.

6 Conclusion

In this paper, we have proposed a hybrid genetic algorithm to solve a three dimensional bin packing problems (3D-BPP) with rotation. The algorithm uses a diploid representation of chromosome to represent the order and rotation values. A modified variant of the DBLF algorithm in terms of residual space calculation is used to realize a 3D packing of the boxes. The GA uses a modified order crossover operator and a two step mutation operator. The generated solutions are optimized using the NSGA II algorithm. The calculated values of average volume utilization, average number of boxes and average value of packed boxes over the entire data-set has been tabulated for no-rotation, 2way rotation and 6way rotation cases. The values show that our proposed algorithm seems to be a good optimizer for 3D-BPP and that 6 way rotation provides better results. For the future work, it is straightforward to examine the performance of our proposed algorithm for more test instances of 3D-BPP and optimize the time complexity of the algorithms used.

References

[1] H. Wang, S. Yang, W. H. Ip, and D. Wang. *Adaptive primal-dual genetic algorithms in dynamic environments.* IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics, 39(6): 1348-1361, 2009.

[2] S. Yang, H. Cheng, and F. Wang. *Genetic algorithms with immigrants and memory schemes for dynamic shortest path routing problems in mobile ad hoc networks.* IEEE Transactions on Systems, Man, and Cybernetics Part C: Applications and Reviews, 40(1): 52-63,2010.

[3] Xueping Li, Zhaoxia Zhao and Kaike Zhang. *A genetic algorithm for the three-dimensional bin packing problem with heterogeneous bins.* Industrial and Systems Engineering Research Conference, 2014

[4] Andrew Lim and Xingwen Zhang. *The Container Loading Problem.* ACM Symposium on Applied Computing, 2005

[5] Andreas Bortfeldt and Hermann Gehring. *A hybrid genetic algorithm for the container loading problem.* European Journal of Operational Research 131 (2001) 143-161

[6] J.A. George and D. F. Robinson. *A Heuristic for packing boxes into a container.* Comput. & Ops Res. Vol. 7. pp. 147-156 Pergamnon Press Ltd., 1980.

[7] Luiz J.P. Araujo, Ajit Panesar, Ender Özcan, Jason Atkin, Martin Baumers and Ian Ashcroft. *An experimental analysis of deepest bottom-leftfill packing methods for additive manufacturing.* International Journal of Production Research, DOI: 10.1080/00207543.2019.1686187

[8] H. Gehring and A. Bortfeldt. *A genetic algorithm for solving the container loading problem.* lnt. Trans. Opl Res. Vol. 4, No. 516, pp.401418, 1997

[9] Sudhansu Mishra, Sukadev Mehera and Gitishree Panda, Ritanjali Majhi. *Portfolio management assessment by four multiobjective optimization algorithm.* Conference Paper · September 2011 DOI: 10.1109/RAICS.2011.6069328

[10] Xi Fang , Wenwen Wang , Lang He, Zhangcan Huang, Yang Liu and Liang Zhang. *Research on Improved NSGA-II Algorithm and Its Application in Emergency Management.* Mathematical Problems in Engineering Volume 2018, Article ID 1306341

[11] Kyungdaw Kang, Ilkyeong Moonb, Hongfeng Wang. *A hybrid genetic algorithm with a new packing strategy for the three-dimensional bin packing problem.* Applied Mathematics and Computation 219 (2012) 1287–1299

[12] Korhan Karabulut and Mustafa Murat Inceoglu. *A Hybrid Genetic Algorithm for Packing in 3D with Deepest Bottom Left with Fill Method.* T. Yakhno (Ed.): ADVIS 2004, LNCS 3261, pp. 441–450, 2004.

[13] H. Monsef, M. Naghashzadegan, A. Jamali, R. Farmani. *Comparison of evolutionary multi objective optimization algorithms in optimum design of water distribution network* Ain Shams Engineering Journal 10 (2019) 103–111

[14] Yanira Gonzaleza, Gara Mirandaa and Coromoto Leona. *Multi-objective Multi-level Filling Evolutionary Algorithm for the 3D Cutting Stock Problem.* Procedia Computer Science 96 (2016) 355 – 364

[15] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. *A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II.* IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, VOL. 6, NO. 2, APRIL 2002

[16] Rio G. L. D'Souza, K. Chandra Sekaran, and A. Kandasamy. *Improved NSGA-II Based on a Novel Ranking Scheme.* JOURNAL OF COMPUTING, VOLUME 2, ISSUE 2, FEBRUARY 2010, ISSN 2151-9617

[17] Joshua Knowles and David Corne. *The Pareto Archived Evolution Strategy : A New Baseline Algorithm for Pareto Multiobjective Optimisation.* 0-7803-5536-9/99/ 1999 IEEE

[18] Hongfeng Wang and Yanjie Chen. *A hybrid genetic algorithm for 3D bin packing problems.* 978-1-4244-6439-5/10/2010 IEEE

[19] J. Knowles and D. Corne. *The Pareto archived evolution strategy: A new baseline algorithm for multiobjective optimization,* IEEE Press, 1999, pp. 98–105.

[20] N. Srinivas and K. Deb. *Multiobjective function optimization using nondominated sorting genetic algorithms.* Evol. Comput.,vol. 2, no. 3, pp. 221–248, Fall 1995.

[21] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. *A Fast and Elitist Multi-objective Genetic Algorithm: NSGA-II* IEEE Trans. Evol. Comp., vol. 6, no. 2, Apr. 2002, pp. 182-197

[22] H. Fang, Q. Wang, Y. Tu, M.F. Horstemeye. *TAn efficient non-dominated sorting method for evolutionary algorithms* IEEE Trans. on Evolutionary Computation, Vol. 16, Issue 3, Fall 2008, pp. 355-384, 2008.

[23] K. D. Tran. *An Improved Non-dominated Sorting Genetic Algorithm-II (ANSGA-II) with adaptable parameters* Jour. of Intelligent Systems Technologies and Applications, Vol. 7, No. 4, Sept 2009.