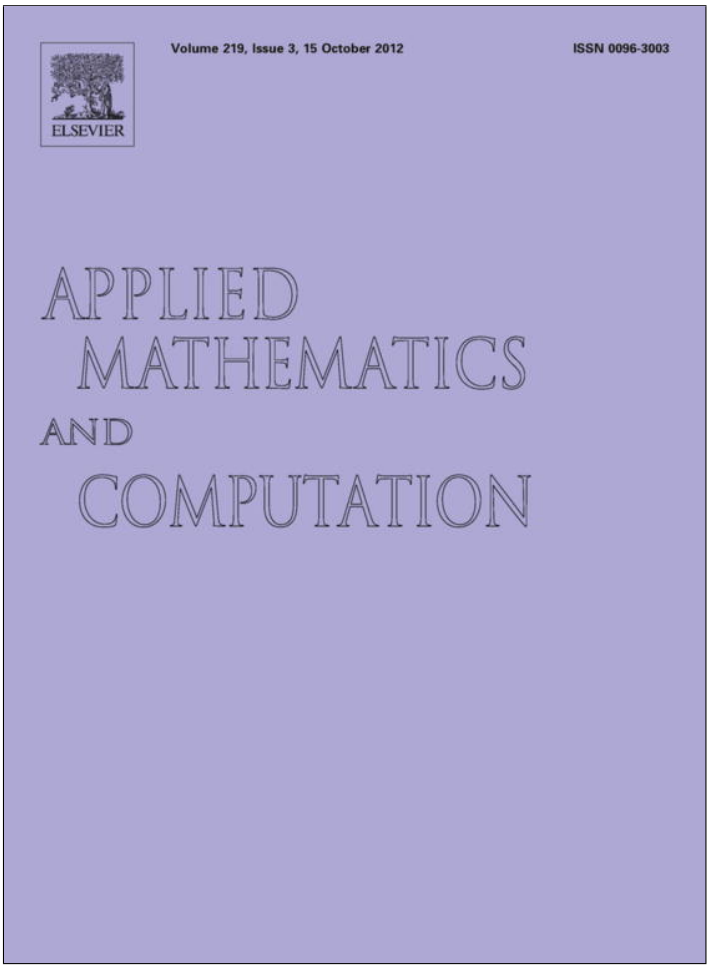

Applied Mathematics and Computation 219 (2012) 1287–1299

Contents lists available at SciVerse ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

A hybrid genetic algorithm with a new packing strategy for the three-dimensional bin packing problem

Kyungdaw Kang [a], Ilkyeong Moon [b,*], Hongfeng Wang [c]

[a] Department of Port Logistics, Tongmyong University, Busan, Republic of Korea
[b] Department of Industrial Engineering, Seoul National University, Seoul, Republic of Korea
[c] College of Information Science and Engineering, Northeastern University, Shenyang, PR China

ARTICLE INFO

Keywords:
Optimization
Bin packing problem
Container loading problem
Genetic algorithm
Heuristic algorithm

ABSTRACT

The three-dimensional bin packing problem is a practical problem faced in modern industrial processes such as container ship loading, pallet loading, plane cargo management, and warehouse management. This article considers a three-dimensional bin packing problem in which objects of various volumes are packed into a single bin to maximize the number of objects packed. The bin packing problem initially appears fairly simple; however, scholars have found its behavior rather complex. The formulated packing strategy seeks to minimize the number of cuboid spaces generated during the packing process by matching the object's height, length, or width with the dimensions of the packing space. A hybrid genetic algorithm was used to solve the three-dimensional bin packing problem with this packing strategy. The results under the proposed algorithm were compared to various sets of computational results found in academic papers related to this subject.

1. Introduction

The three-dimensional Bin packing problem (3D-BPP) directly models several issues in production and transportation planning; as such, it has many practical applications in various fields such as container ship loading, pallet loading, plane cargo management, and warehouse management. The effective utilization of materials and transportation capacity is a major competitive advantage for enterprises. The modern economy is driven by multinational conglomerates whose advantage lies in the transcontinental shipment of large amounts of materials and products. This has led to research into the efficient management of the supply chain as a means to reduce costs. To this end, the pursuit of a useful packing method remains an important field in academia because of its extensive real world applications. The large-scale effects of even a relatively incremental growth in the utilization of materials and space capacity in modern production and distribution processes can yield a considerable benefit to the supply chain. Another crucial benefit of minimizing the wasted volume of a container is that densely packaged boxes decrease the chance of breakage during shipment.

This article presents a Hybrid genetic algorithm (HGA) method for the 3D-BPP, which is suitable for a single container packing strategy for both homogeneous and heterogeneous box sets. The packing strategy introduced in this study is a variant of the deepest bottom left packing method developed by Karabulut and Mustafa [1]. Three box orientations were considered, with experiments conducted using a modified version of the Bischoff/Ratcliff data set generator. Heuristic algorithms used to solve the 3D-BPP can generally be categorized into three groups: conventional heuristics, metaheuristics, and tree or graph searches. A conventional heuristic algorithm generally utilizes a packing strategy based on the greedy algorithm and

* Corresponding author.
E-mail addresses: jay.kang@tu.ac.kr (K. Kang), ikmoon@snu.ac.kr (I. Moon), hfwang@mail.neu.edu.cn (H. Wang).

0096-3003/$ - see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.amc.2012.07.036

improvement methods similar to local search algorithms. Efficient conventional heuristics for solving the 3D-BPP were suggested by Bischoff and Ratcliff [2], Bischoff and Ratcliff [3] and Lim et al. [4]. Over the last 10 years, metaheuristic search strategies have constituted the preferred approach to the 3D-BPP. Numerous metaheuristic search strategies have been introduced by Hemmink [5] and Bortfeldt and Gehring [6–8]. Notable uses of metaheuristic search strategies include Karabulut and Mustafa's [1] use of genetic algorithms with the Deepest Bottom Left with Fill (DBLF) and Liang et al.'s [9] proposed two-phased Ant Colony Optimization (ACO).

In literature, researchers generally use some form of layering technique for 3D bin packing in which a bin is packed layer by layer. However, layering algorithms have the drawback of performance degradation if the input forces fragmentation of the loading surface during the early stages. The heuristic packing strategy proposed by Karabulut and Mustafa [1] follows a long line of Bottom Left (BL) and Bottom Left with Fill (BLF) methods which have been used in 2D bin packing. Baker et al. [10] first described a bottom-up left-justified heuristic for 2D packing, where boxes are placed one by one and the current box is placed at the lowest, left-most position into which the box fits. Research on 2D bin packing has witnessed numerous improvements since the 1980s, with Hopper [11] introducing a 2D based packing strategy for the BLF algorithm, in which 2D objects are arranged in the lowest possible region of the bin to fill the empty spaces in the layout. Karabulut and Mustafa [1] proposed the DBLF, an extension of Hopper [11]'s BLF heuristic to three dimensions, where an object is moved to the deepest available position (smallest z value) in the layout, then as far as possible to the bottom (smallest y value), and finally as far as possible to the left (smallest x value). A major disadvantage of Hopper [11]'s algorithm, as stated by Karabulut and Mustafa [1], is its complexity.

Generally, academic research on 3D-BPP has focused on developing and refining a packing strategy. It can be noted that a great majority of packing strategies feature overlap and orientation constraints as well as further constraints on the weight of the container. The heuristic developed by Bischoff [12] for loading pallets and the GRASP method from Moura and Oliveira [13] prove to be particularly sensitive methods for realizing container stability. Bischoff [12] also considered a complex overstacking constraint, which limits the pressure that is applied to the boxes in a stack. Additional constraints on bin packing problems lead to a class of problems named the Container loading problem (CLP). In CLP, all containers have fixed dimensions, and all boxes are to be packed into the minimum number of containers. A complex combination of KP and the BPP is the Multi Container Packing Problem (MCPP), in which a container may have varying dimensions and the objective is to choose a subset of containers that results in the lowest total shipping cost. To date, only orthogonal objects have been considered, but polygonal figures are also an area of research that has potential benefits for the steel and textile industries. The problem is a combinatorial optimization problem that focuses on how to achieve optimal space utilization in a limited space. The given objects are arranged according to respective constraints and characteristics to achieve effective space utilization and maximize the space utilization ratio.

2. Problem description

2.1. Problem definition and constraints

In this paper, a hybrid genetic algorithm is used to solve the three-dimensional bin packing problem. The constraints for the 3D-BPP are as follows. First, each box lies completely in the container and does not penetrate the container's boundary surfaces. Second, no two packed boxes can overlap within the container. Finally, each box is parallel to the container's surfaces and is supported by either the floor of the container or other packed boxes. The cuboid container consists of dimensional values, and the space within the boundary surface is represented by a coordinate vector (position). Fig. 1 illustrates this container and introduces terms such as "deep", "bottom", and "left". The origin of the x, y, and z axes represents the deepest bottom left coordinate in the container.

In the algorithm, a unique instance of a box consists of three values for dimension and will be arranged in the order prescribed by the packing strategy. A box packed within the container is assigned three values regarding position and remains immovable in any instance of the packing strategy. While such a layering technique provides immediate results for 3D bin packing algorithms, early fragmentation of the packed boxes will result in performance degradation. Scholars have recommended a greater variation of packing solutions and diversity of box types as methods of mitigating this inherent weakness. This paper considers the constraint of box rotation for greater variation of box types. Fig. 2 illustrates the six possible cases for box rotation, with case 0 representing a box with no rotation.

This paper considers three possible box orientations: no rotation; x-axis (2-way) rotation; and three axes (6-way) rotation. 3D-BPP represents an important scholarly research topic and is a combinatorial optimization problem in operations research. An algorithm designed for 3D bin packing must consider the stated constraints and automate the packing process so that it can be applied easily to different variants of the problem. The algorithm must ensure that there is no overlap between the boxes that are inserted into the container. According to the existing literature, in terms of the loading of objects in space, there are primarily four kinds of packing situations: single- container and multiple- container packing and the packing of the same and different box types. The single- and multiple- container packing situations are either maximization or minimization problems. In the maximization problem, the objective is to pack the maximum number of boxes into a single container. In the minimization problem, the objective is to pack all the boxes into the minimum number of multiple containers; the two box types are defined by the variety of its dimension. If there is a generic box type, the problem is referred to as

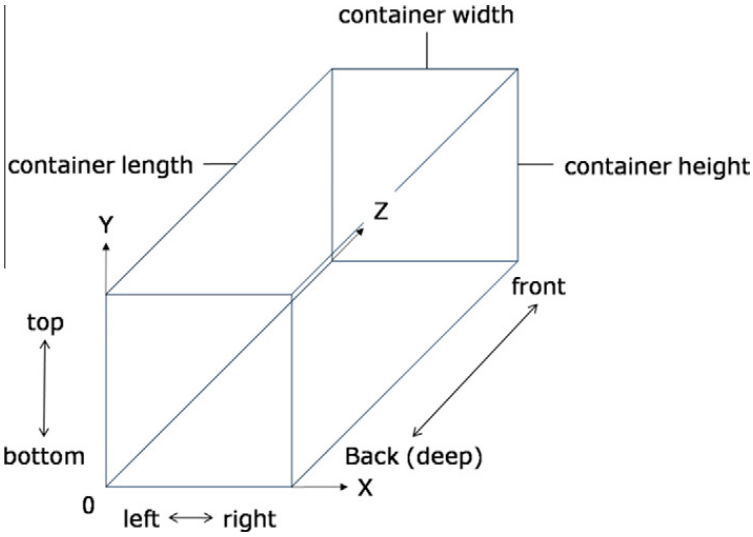

Fig. 1. Container in a 3D coordinate system.

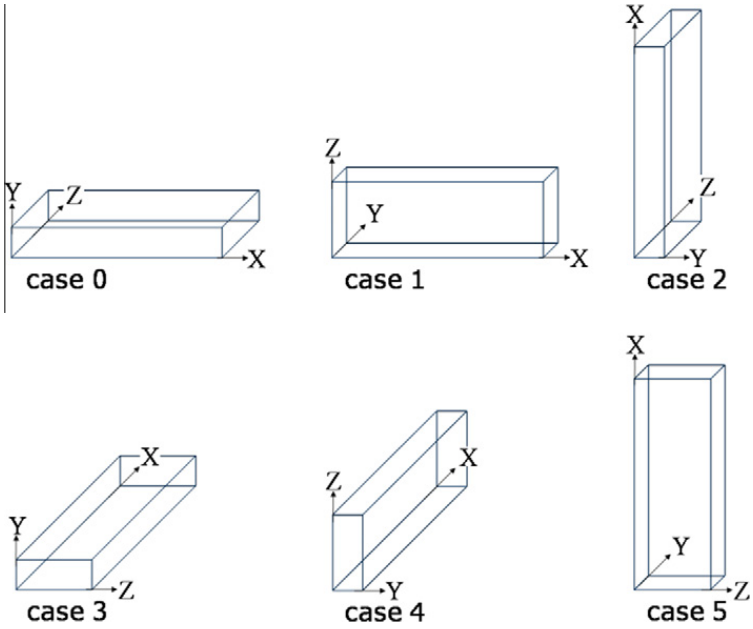

Fig. 2. Six cases of box rotation. Case 0: No rotation, Case 1: Rotate the box around the y-axis by 90°, Case 2: Rotate the box around the *z*-axis by 90°, Case 3: Rotate the box around the *x*-axis by 90°, Case 4: Rotate the box around the *x*-axis by 90° and then around the *z*-axis by 90°, Case 5: Rotate the box around the *z*-axis by 90° and then around the *x*-axis by 90°.

homogeneous. If there are only a few box types with a relatively large number of specimens per type, the box set is weakly heterogeneous; likewise, if there are various box types with only a few similarities per type, the box set is strongly heterogeneous. The importance of defining the problem by the box type arises because various algorithms work best for either homogeneous or heterogeneous boxes.

2.2. Cuboid space definition

The cuboid space into which a box can be placed is the packable area within the container used by the proposed metaheuristic methods. The I-DBLF method stores both the dimensional and positional values for the space object. Fig. 3 shows the origin with a single box placed at the deepest bottom left position.

During initialization, a coordinate vector (space list) is created with a single space object in the list. The initial space has the same dimensional value as the container, with the origin as its position. Throughout the packing process, a box is inserted into a space; this results in new space objects due to the displaced area. The maximum number of spaces generated from a single packing process is three. Fig. 4 illustrates these space objects as a continuation of the previous figure; these objects are represented as the "side", "top", and "front" overlapping packable spaces. A space is removed once a box is packed within it or if the space is detected as being infeasible.

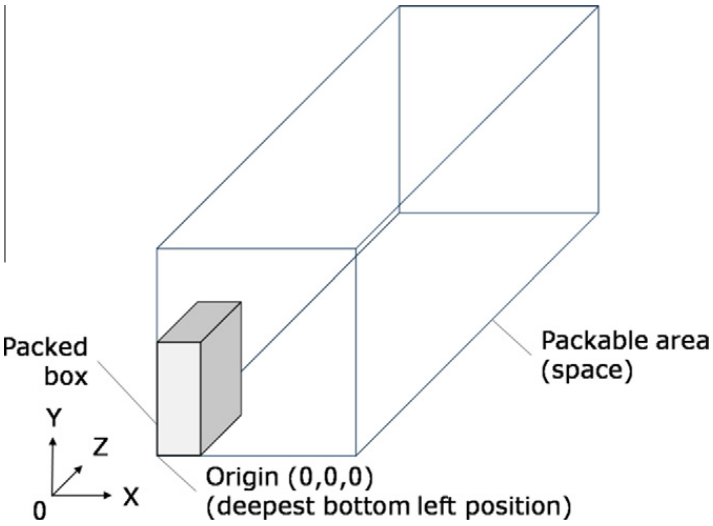

Fig. 3. The allowable packing area within a container.

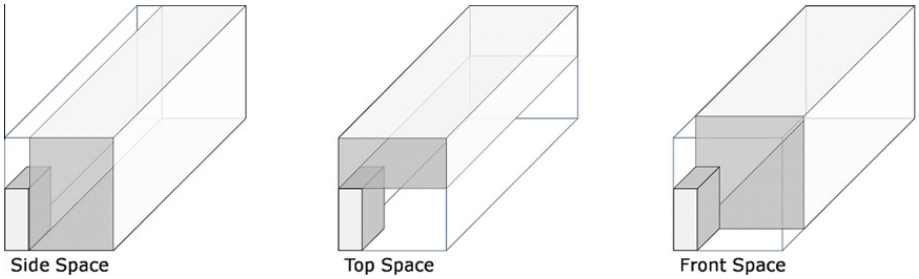

Fig. 4. Three spaces displaced by the inserted box.

The unassigned box list stores box objects that have not been packed into a container. A box in the unassigned box list has negative values for its position. This list is used in the I-DBLF to find the box volume that is currently smallest. This value is used to mark infeasible space objects and remove them from the space list. When a box has met all the constraints and is packed into the container, the position is obtained from the selected space, relocated from this list, and appended to the assigned box list.

The packed box list is either an incomplete or complete solution to the given problem instance. A solution is complete when all the unassigned boxes within the list have been attempted to be inserted into the container. Fig. 5 depicts a container with a width of 10, a height of 10, and a length of 10 and all twenty boxes packed, as sequenced in Table 1. The utilization rate is the total volume of the boxes within the packed box list and represents the performance level of the solution,

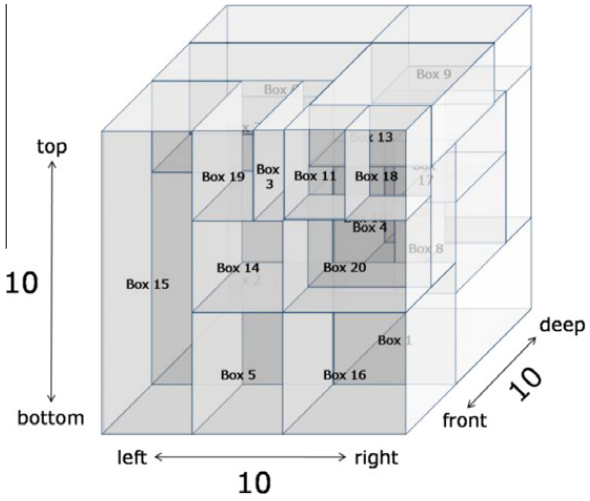

Fig. 5. Packing layout for a container with 100% utilization.

Table 1
Packing sequence for a container with 100% utilization.

#	Dimension			Position			R	Volume	#	Dimension			Position			R	Volume
2	6	7	6	0	0	0	1	252	14	3	3	4	3	4	6	3	36
1	4	3	6	6	0	0	2	72	8	2	3	4	8	3	2	2	24
12	4	5	2	6	3	0	3	40	13	4	2	5	6	8	3	5	40
10	2	5	4	6	3	2	4	40	4	4	4	2	6	4	6	2	32
15	3	10	4	0	0	6	0	120	19	2	3	4	3	7	6	3	24
6	6	3	3	0	7	0	3	54	3	1	3	4	5	7	6	1	12
5	3	4	4	3	0	6	3	48	20	4	3	2	6	4	8	2	24
16	4	4	4	6	0	6	3	64	17	2	2	4	8	6	2	2	16
7	6	3	3	0	7	3	3	54	11	2	3	2	6	7	8	2	12
9	4	2	3	6	8	0	3	24	18	2	3	2	8	7	8	5	12

Best Generation in 951 with fitness value of **100%** and **20** boxes packed.

with greater values having precedence. Each box within the packed box list has met the constraints of the problem and has values for the rotation and the position.

3. Genetic algorithm

Genetic algorithms (GA) are adaptive methods that have gained wide acceptance for solving search and optimization problems. The principle of evolution of biological organisms is the main idea behind genetic algorithms and was first introduced by Holland [14] in 1975. Based on the simulation of natural evolution, the building blocks of a GA are "chromosomes", which represent the candidate solutions. The GA evolves through an iterative process in which each individual chromosome is given a chance to "reproduce" based on the assigned "fitness score." All chromosomes are given the chance to reproduce; however, fitter individuals have a greater chance for selection. Reproduction is performed in the population through a process called "crossover". A crossover produces new children (or offspring) that inherit some features from each parent. Because less fit members have smaller chances of reproducing, they fade away from the population during evolution. After recombination, a mutation operator can be applied. Each bit in the population can be mutated (flipped) with a low probability that is generally smaller than one percent. A good GA will converge to an optimal solution after several iterations. This basic implementation of a GA is not guaranteed to find the global optimum solution to a problem, but it is generally good at finding "acceptably good" solutions to problems "acceptably quickly". Because GAs are general methods, they do not contain valuable domain information; thus, hybridization with domain information can dramatically increase the quality of the solution and the performance of a genetic algorithm through smarter exploration of the search space.

In this research, a solution is encoded as a list of n boxes, which have been assigned to chromosome $b = (b_1,(ab_n)$ in a unique space in the container. The GA encoding scheme is based on Raidl and Kodydek's [15] Order Based Encoding (OBE), which was introduced in "Genetic Algorithms for the Multiple Container Packing Problem". Fig. 6 shows the GA encoding scheme for the proposed metaheuristic method.

OBE is used to explore the search space, and each encoded chromosome represents a feasible solution to the 3D-BPP. The packing strategy is a single-pass type heuristic, and if all boxes have been tried, the remaining boxes are treated as unassigned. OBE ensures the improvement of the GA by attempting to repack the unassigned boxes during the local search heuristic. The population is evolved to find as many distinct box packing sequences as possible. The initial population has six chromosome types; the first is an unaltered chromosome, next is a box sequence that is randomly generated, and the last four chromosome types are based on the box sequence that is sorted in descending order of the volume, width, height, and length. Later chapters will further discuss the impact of population diversity in 3D-BPP with GA.

4. Packing strategy

4.1. Simple deepest bottom left with fill

The Simple Deepest Bottom Left with Fill (S-DBLF) packing method is based on Karabulut and Mustafa's [1] DBLF algorithm. The S-DBLF procedure tries to pack each box into the first allowable DBL position; its pseudo code is shown in Fig. 7.

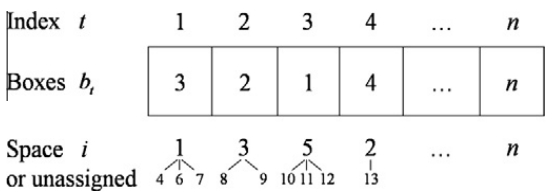

Fig. 6. A GA coding scheme based on OBE.

```
Procedure S-DBLF packing method:
begin
        initialize the position set P;
        for i:= 1 to |L| do
        boo:=false;
        sort the position set P according to the DBL order;
        for j:=1 to |P| do
            if box i ∈ L can be packed into position j ∈ P then
                boo:=true;
                set position j as the packed position of box i;
                update the position set P;
                break;
            end if
        end for
        if boo==false then break;
        end for
end;
 Denotations:
 L: a packed sequence list for a set boxes
```

Fig. 7. Pseudo-code for the S-DBLF packing method.

The pack strategy begins with the container's position set P being initialized with coordinates (0, 0, 0) as the origin. For each box packed from the box list L, the set P is sorted in the deepest bottom left order. The selected box i will be verified to see if packing into position j in set P is feasible. The algorithm includes two checks: first, it is verified that the box does not penetrate the boundary surfaces of the container. Second, it is verified that the box does not overlap with previously packed boxes. If box i can be packed into position j, box packed position is set as j and set P is updated by deleting position j and adding three new positions representing the displaced area by packed box. If box i cannot be packed into position j, the next box in list L is considered. If all positions in P have been tried, the packing operation will terminate. The S-DBLF is a single-pass type heuristic that relies on GA to find the best possible solution and is introduced to compare with the I-DBLF algorithm. The results for the S-DBLF packing strategy with 100 strongly heterogeneous boxes without rotation are shown in Table 2.

Table 2
Results for S-DBLF with no rotation.

No	Dimension			Position			Volume	no	Dimension			Position			Volume
53	81	147	90	0	0	0	1071,630	35	112	125	88	0	94	948	1232,000
48	125	123	60	81	0	0	922,500	17	233	88	90	0	121	614	1845,360
19	51	72	59	81	123	0	216,648	23	53	44	36	172	159	352	83,952
18	67	42	54	132	123	0	151,956	2	40	53	44	172	159	388	93,280
21	46	64	66	0	147	0	194,304	52	80	114	169	148	0	769	1541,280
38	233	235	262	0	0	90	14,345,810	15	117	65	163	100	0	549	1239,615
28	54	30	69	132	165	0	111,780	26	64	114	89	163	121	492	649,344
24	166	121	133	0	0	352	2671,438	14	170	170	117	0	0	1080	3381,300
33	37	26	24	186	165	0	23,088	12	70	65	117	163	0	948	532,350
20	64	48	32	166	0	352	98,304	39	97	69	60	72	164	883	401,580
3	172	83	81	0	121	352	1156,356	16	112	101	125	112	94	948	1414,000
47	163	101	181	0	121	433	2979,803	30	54	40	63	166	0	384	136,080
51	100	121	230	0	0	485	2783,000	49	63	117	116	170	0	1080	855,036
57	72	181	233	0	0	715	3036,456	41	36	44	15	186	191	37	23,760
56	123	125	54	72	0	715	830,250	27	72	44	112	100	65	549	354,816
55	54	67	27	81	0	60	97,686	60	46	61	81	166	48	400	227,286
44	76	164	114	72	0	769	1420,896	1	83	64	37	72	125	715	196,544
54	163	94	132	0	0	948	2022,504	4	37	61	81	170	117	1080	182,817
13	45	38	37	186	191	0	63,270	6	64	66	29	135	0	60	122,496
22	155	68	114	72	164	769	1201,560	45	31	72	24	46	147	0	53,568
42	80	114	64	100	0	485	583,680	43	44	36	38	186	191	52	60,192
29	69	114	59	163	121	433	464,094	9	26	37	42	206	0	0	40,404
36	38	83	25	81	123	59	78,850	46	37	26	17	186	165	24	16,354
25	72	32	59	81	195	0	135,936	8	31	59	72	0	170	1080	131,688
40	43	64	48	166	48	352	132,096	10	38	37	20	132	123	54	28,120
31	46	38	65	172	121	352	113,620	34	54	77	63	72	0	883	261,954

Best fitness value of **79.16%** and **52** boxes packed.

4.2. Improved deepest bottom left with fill

The proposed heuristic method referred to as improved deepest bottom left with fill (I-DBLF) uses a hybrid genetic algorithm to solve the 3D-BPP. GAs have proven to be very well suited to finding nearly optimal solutions to combinatorial problems. The I-DBLF algorithm presents a packing strategy for hybridizing the genetic algorithm to convert the 3D-BPP into a common optimization problem. The proposed heuristic packing strategies follow Hopper [11]'s BLF heuristic of first placing the box at the deepest available position in the layout, then moving it as far as possible to the bottom, and finally moving it as far as possible to the left. The I-DBLF method introduces cuboid space objects as the allowable packing area within the container for computational efficiency. The space objects are used to compare the feasibility of box insertion before implementing computationally intensive processes, such as checking for intersections and local search methods. I-DBLF's packing strategy performs an additional iteration of the list of spaces after the initial packing area is selected. Conditions for removing infeasible space objects are applied for optimization. Fig. 8 shows a flowchart of the I-DBLF packing strategy.

The data are first initialized in preparation for the packing strategy. First, a space object equal in volume and dimension to the container is added to the list. Then, a common strategy, as noted in Fanslau and Bortfeldt [16], of sorting the initial *unassigned box list* in descending order by box type is utilized. As stated in the previous chapter, five data filters are randomly applied in this work. The smallest volume size in the box sequence is initialized and updated as boxes are packed within the container.

The main iteration of the packing strategy repeats until all boxes in the *unassigned box list* have been tried. A box is selected sequentially and must be packed in the DBL position within the space available in the list. To arrange a box within a space, two conditions must be satisfied: first, it must have a volume larger than that of the box, and second the space dimensions must be larger than the box dimensions. A space that is not able to satisfy these two conditions is abandoned for the next space until all spaces in the list have been tried. The intersection test with all the boxes in the *assigned box list* is a computationally heavy process, and these conditions ensure that no unnecessary calculations are performed. If no intersection is detected, the box is packed in the selected space; this process is illustrated in Fig. 9.

The proposed box insertion strategy ensures that the DBL position is found and triggers the following: the box is removed from the *unassigned box list* and added to the *assigned box list*. The selected space is removed from the list; the values representing the displaced *width*, *height* and *length* relative to the packed position minus the container's dimension illustrated in Fig. 4 are used to calculate the new space object's dimensions. For a space object to be created, its volume must be larger than or equal to the smallest box volume in the *unassigned box list*. A maximum of three spaces can be generated by the box insertion process, which represents the "side", "top", and "front" areas within the container. The space objects are created relative to the container size and within its list repacked into better position. First, the temporary positions of x, y, and z of negative value are created. This is repeated for the length of a space array. The next space from the space list is evaluated. If the number of selected space positions is less than the number of space positions in the list and if no intersection is detected for a position, the new selected position of the dimension is set. These cases represent the weakness in the DBL method and are analyzed in the case study discussed in this article. The selected box and space are parameters for this method with the main iteration for the size of the space list. If the positional values of the space objects in the list are smaller than the positional values of the selected space, an intersection test is executed. If no overlap is found, the new position is stored and used to

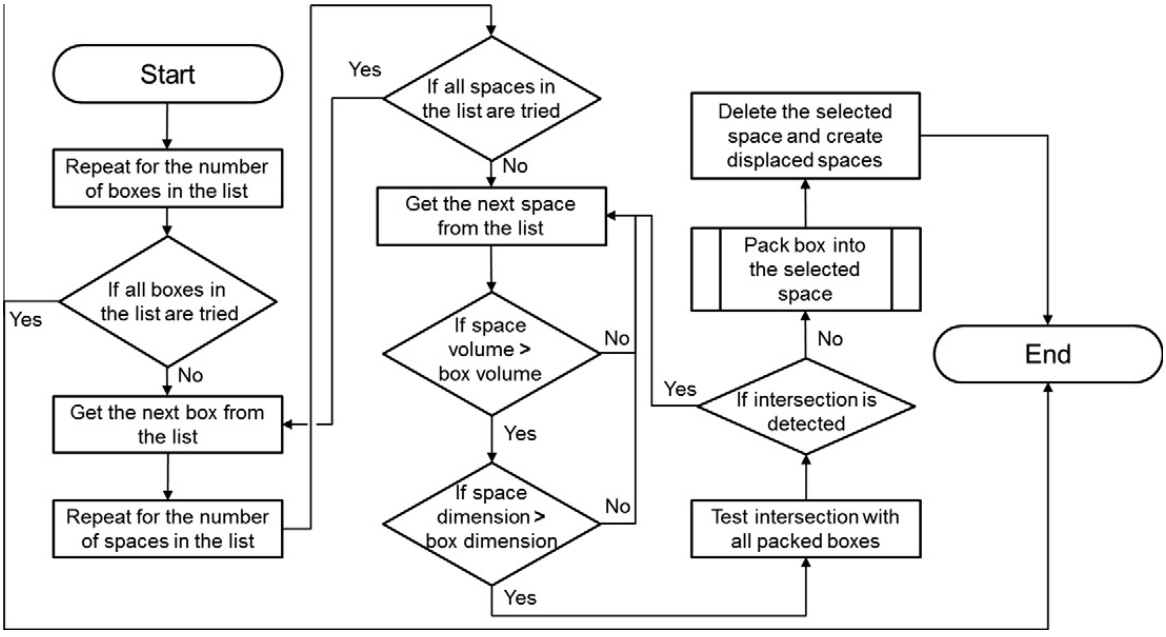

Fig. 8. Flowchart of the I-DBLF packing strategy.

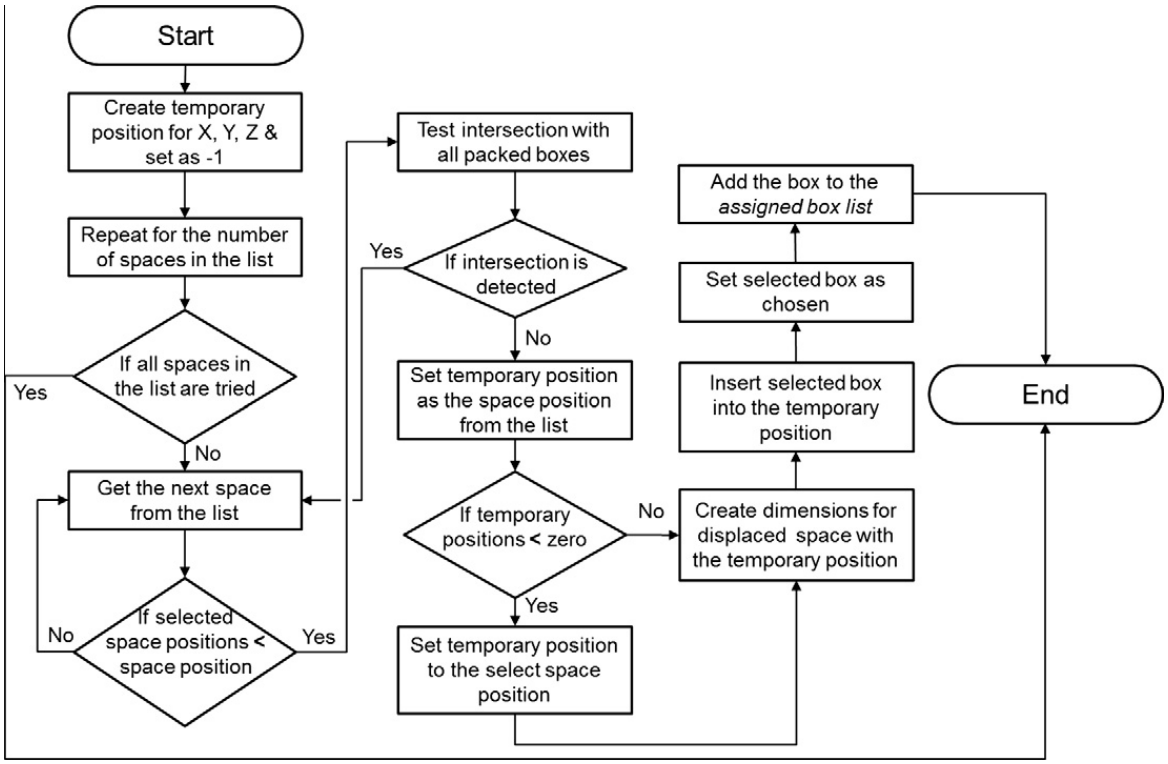

Fig. 9. Flowchart of the I-DBLF box insertion.

find the smallest feasible position for each axis. If no space object in the list is less than the selected space position, the original selected position is set. Once the insertion process has been completed, the position list is updated and evaluated to remove any unnecessary positions. In the final step, the new space list is sorted in the deepest bottom left order. Then, the next box is taken from the list, and the same selection procedure is repeated. The box insertion we have proposed solves deviant cases in which finding the DBL position is difficult and requires additional computation, as illustrated in Fig. 9.

5. Case study

5.1. Weakness in DBL methods

The following case study was illustrated by Zhu [18] in 2009; it proposed that previous bottom left heuristics does not optimally find the deepest bottom left position given certain circumstances. The blog post provides a dataset generated based on Bischoff and Ratcliff [2] and an analysis of previous researched by Karabulut and Mustafa [1], Baker [10], Hopper [11] and Jakobs [17]. It has been stated that both Hopper [11] and Karabulut [1] failed to place the boxes at the deepest possible position. In this study, a container with a width of 3, a height of 3, and a length of 4, as shown in Fig. 10, has five boxes packed as sequenced in Table 3.

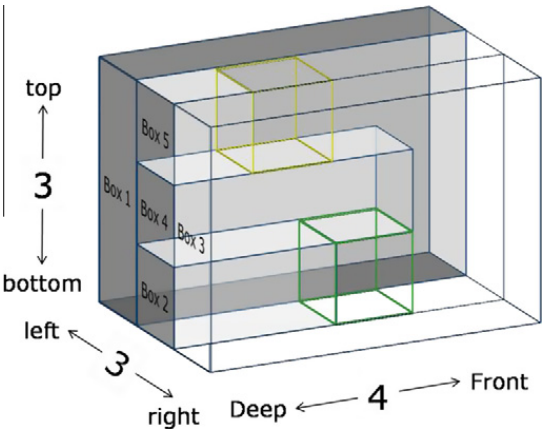

Fig. 10. An exception in which the DBL position is not found by previous research.

Table 3
An exception where the DBL position is not found by previous research.

	Dimension			Position		
	Width	Height	Length	*X*	*Y*	*Z*
Box 1	1	3	4	0	0	0
Box 2	1	1	2	1	0	0
Box 3	1	3	4	2	0	0
Box 4	1	1	3	1	1	0
Box 5	1	1	1	1	2	0
Box 6	1	1	1			

According to previous bottom left heuristics, Box 6 will start at the position (2, 2, 4), move downward to (2, 0, 4), and next move to the left to (1, 0, 4), before moving deeper to (1, 0, 2) and sitting behind Box 2 in the green position. In fact, the deepest position should be the yellow position, (1, 2, 1), next to Box 5. Karabulut and Mustafa's [1] DBLF method could not find this position, but our packing strategy was able to locate the optimal deepest bottom left location.

The I-DBLF locates the correct deepest bottom left position by utilizing space objects. Space object creation is a notable contribution of this strategy and is illustrated in Fig. 11. Initially a space object with dimensions equal to those of the container is added to the list (*Step 1*). Box 1 is inserted into the position of origin at (0, 0, 0); because the height and length of Box 1 are the same as those of the container, no space object is created for these areas. The side space is generated by subtracting the width of Box 1 from the width of the container (*Step 2*). Space 1 is then added to the list, and the initial selected space 0 is deleted. Box 2 is inserted into this newly created space, and three space objects are created with overlapping dimensions (*Step 3*); space 1 is then removed. Next, space 4 is selected for Box 3, which has equal dimensions, and no space objects are generated (*Step 4*). Box 4 is arranged within space 3, and three space objects are generated accordingly (*Step 5*). Finally, the previously generated space 6 is selected for Box 5, and two space objects are generated for the deepest and side position (*Step 6*).

The last box to be inserted can be arranged either in space 2 or space 8, which are represented by the yellow and green positions in Fig. 10, with space 2 representing the deepest position within the deepest bottom left heuristics.

5.2. Effects of sorting of initial population

According to scholars, such as Raidl and Kodydek [15], the sorting of the initial generation greatly improves GA performance. In this proposed algorithm, the boxes were sorted by their volumes and dimensions. The effects of the sorting were examined by 100 instances of sorted and unsorted solutions, as shown in Table 4.

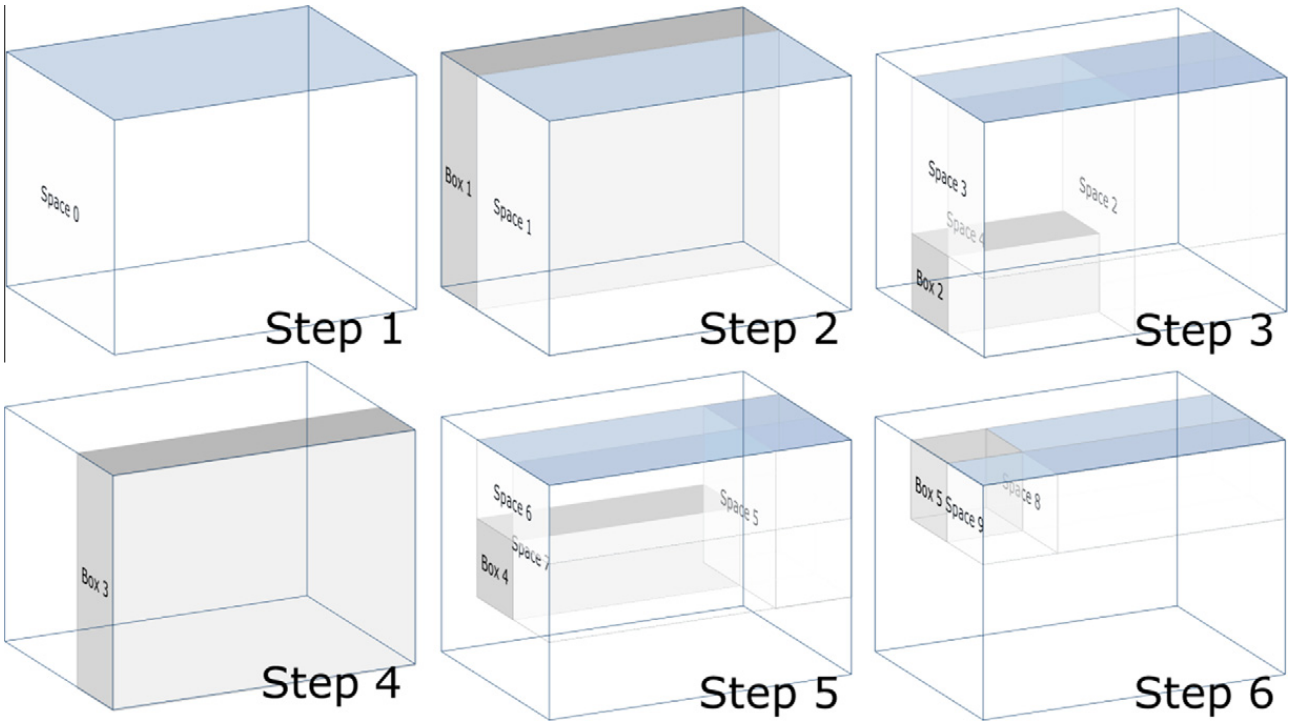

Fig. 11. Space creation process for I-DBLF box insertion.

Table 4
Comparison of results for the sorted and unsorted I-DBLF.

	SGA	No Rotation	2-Way	6-Way
I-DBLF w. sort	79.16%	86.30%	90.64%	94.78%
I-DBLF no sort		48.89%	85.23%	88.31%

Table 5
Statistical analysis on the effects of sorting initial population.

	Average	SD	Min	Max
I-DBLF w. sort – 6W	89.44%	1.75	85.32%	93.91%
I-DBLF no sort – 6W-Homogeneous	78.75%	0.66	75.42%	79.29%
I-DBLF no sort – 6W-Heterogeneous	85.21%	8.08	37.20%	88.31%

This initial population improves the packing process and results in a higher utilization rate. The most dramatic change can be found in boxes that have no rotation. Table 5 examines the test case with all rotations to show the effects on sorting an initial population into strongly heterogeneous boxes and homogeneous boxes.

The analysis shows that sorting stabilizes homogeneous boxes, resulting in a standard deviation value of .66, whereas heterogeneous boxes have the opposite effect. This difference can be attributed to the fact that some heterogeneous boxes have dimensions equal or close to those of the container; therefore, if these boxes are not packed first by sorting by volume, they may never be assigned within the algorithm. Meanwhile, homogeneous boxes suffer more from the lack of diversity in the population and concentrate the utilization in the results found in Table 4 for SGA.

6. Computational experiments

Tables 6–8 show the packing solutions for the I-DBLF method introduced in this study. The detailed sequences are as follows: Table 6 displays the results with no rotation, Table 7 shows the results with two-way rotation (rotation allowed only around the *x*-axis) and Table 8 shows the results with six-way rotation (all rotations allowed).

The computational results for the I-DBLF method were evaluated with regard to the utilization rate of the container volume. The Bischoff and Ratcliff [2] dataset generator ensures that a utilization rate of 100% is possible. The experiments were

Table 6
Results for I-DBLF with no rotation.

No	Dimension			Position			*R*	Volume	no	Dimension			Position			*R*	Volume
38	233	235	262	0	0	0	0	14,345,810	37	64	121	133	166	81	791	0	1029,952
14	170	170	117	0	0	262	0	3381,300	17	233	88	90	0	125	1100	0	1845,360
57	72	181	233	0	0	379	0	3036,456	1	83	64	37	114	123	1059	0	196,544
47	163	101	181	0	0	612	0	2979,803	19	51	72	59	172	0	442	0	216,648
54	163	94	132	0	101	612	0	2022,504	21	46	64	66	172	0	501	0	194,304
24	166	121	133	0	101	744	0	2671,438	4	37	61	81	163	147	612	0	182,817
51	100	121	230	72	0	379	0	2783,000	18	67	42	54	0	181	379	0	151,956
5	233	81	152	0	0	793	0	2868,696	46	37	26	17	155	178	262	0	16,354
58	114	165	155	0	0	945	0	2915,550	33	37	26	24	192	178	262	0	23,088
52	80	114	169	72	121	379	0	1541,280	41	36	44	15	152	178	279	0	23,760
59	90	164	114	114	0	945	0	1682,640	10	38	37	20	188	178	286	0	28,120
7	119	123	125	114	0	1059	0	1829,625	9	26	37	42	142	178	294	0	40,404
22	155	68	114	0	165	945	0	1201,560	45	31	72	24	170	117	343	0	53,568
35	112	125	88	0	0	1100	0	1232,000	43	44	36	38	172	77	379	0	60,192
49	63	117	116	170	0	262	0	855,036	13	45	38	37	172	77	417	0	63,270
48	125	123	60	0	81	877	0	922,500	30	54	40	63	0	181	433	0	136,080
32	69	147	90	163	0	612	0	912,870	25	72	32	59	0	181	496	0	135,936
60	46	61	81	170	117	262	0	227,286	40	43	64	48	169	121	548	0	132,096
34	54	77	63	172	0	379	0	261,954	8	31	59	72	200	147	612	0	131,688
27	72	44	112	0	170	262	0	354,816	6	64	66	29	152	121	502	0	122,496
39	97	69	60	72	121	548	0	401,580	31	46	38	65	172	72	454	0	113,620
29	69	114	59	152	121	379	0	464,094	28	54	30	69	172	64	519	0	111,780
12	70	65	117	72	170	262	0	532,350	20	64	48	32	152	187	502	0	98,304
50	74	69	114	155	164	945	0	582,084	55	54	67	27	163	0	702	0	97,686
42	80	114	64	152	121	438	0	583,680	2	40	53	44	0	181	555	0	93,280
26	64	114	89	166	81	702	0	649,344	23	53	44	36	155	178	336	0	83,952
									36	38	83	25	125	81	877	0	78,850

Best Generation in 38 with fitness value of **86.30%** and **53** boxes packed.

Table 7
Results for I-DBLF with 2-way rotation.

No	Dimension			Position			R	Volume	No	Dimension			Position			R	Volume
38	233	235	262	0	0	0	1	14,345,810	39	60	69	97	0	163	961	1	401,580
14	170	170	117	0	0	262	1	3381,300	27	72	44	112	0	182	451	1	354,816
57	233	181	72	0	0	379	1	3036,456	34	54	77	63	169	121	897	1	261,954
47	181	101	163	0	0	451	1	2979,803	60	46	61	81	181	0	451	0	227,286
58	155	165	114	0	0	614	1	2915,550	19	51	72	59	181	0	532	1	216,648
5	233	81	152	0	101	451	0	2868,696	9	42	37	26	170	117	262	1	40,404
51	230	121	100	0	0	728	1	2783,000	45	31	72	24	199	123	961	1	53,568
24	166	121	133	0	0	828	1	2671,438	43	38	36	44	170	117	288	1	60,192
54	132	94	163	0	121	728	1	2022,504	13	45	38	37	170	117	332	1	63,270
11	87	163	132	0	0	961	1	1871,892	36	25	83	38	199	123	985	1	78,850
7	125	123	119	87	0	961	1	1829,625	23	36	44	53	0	181	379	1	83,952
17	233	88	90	0	0	1093	0	1845,360	2	40	53	44	36	181	379	1	93,280
52	80	114	169	132	121	728	1	1541,280	55	27	67	54	60	163	961	1	97,686
44	76	164	114	155	0	614	1	1420,896	20	64	48	32	76	181	379	1	98,304
16	112	101	125	87	123	961	1	1414,000	28	69	30	54	140	181	379	1	111,780
15	163	65	117	0	170	262	1	1239,615	31	65	38	46	72	182	451	1	113,620
22	155	68	114	0	165	614	0	1201,560	6	29	66	64	199	123	1023	1	122,496
53	81	147	90	0	88	1093	0	1071,630	8	72	59	31	0	163	1058	1	131,688
37	64	121	133	166	0	828	0	1029,952	25	59	32	72	137	182	451	1	135,936
32	90	147	69	81	88	1093	1	912,870	30	63	40	54	72	182	497	1	136,080
49	63	117	116	170	0	262	0	855,036	18	54	42	67	135	182	523	1	151,956
26	89	114	64	0	121	891	1	649,344	4	81	61	37	81	88	1162	1	182,817
42	80	114	64	89	121	897	0	583,680	1	83	64	37	81	149	1162	1	196,544
50	74	69	114	155	164	614	1	582,084	10	38	37	20	76	181	411	0	28,120
12	70	65	117	163	170	262	0	532,350	41	15	44	36	212	117	262	1	23,760
29	59	114	69	171	88	1093	1	464,094	33	24	26	37	209	181	379	1	23,088
									46	37	26	17	36	181	423	1	16,354

Best Generation in 54 with fitness value of **90.64%** and **53** boxes packed.

Table 8
Results for I-DBLF with 6-way rotation.

No	Dimension			Position			R	Volume	No	Dimension			Position			R	Volume
38	233	235	262	0	0	0	0	14,345,810	60	61	81	46	170	0	379	4	227,286
14	170	117	170	0	0	262	4	3381,300	34	54	63	77	172	152	667	4	261,954
57	233	181	72	0	0	432	1	3036,456	27	72	44	112	125	181	432	0	354,816
47	101	181	163	0	0	504	4	2979,803	39	97	69	60	88	114	1126	2	401,580
58	165	114	155	0	117	262	1	2915,550	19	59	51	72	0	181	555	5	216,648
5	233	152	81	0	0	667	1	2868,696	1	37	64	83	191	123	1001	1	196,544
51	100	121	230	0	0	748	2	2783,000	21	64	46	66	60	183	1126	2	194,304
24	133	121	166	100	0	748	5	2671,438	4	37	61	81	191	123	1084	0	182,817
54	132	163	94	101	0	504	3	2022,504	18	67	42	54	125	163	544	0	151,956
11	132	163	87	100	0	914	2	1871,892	30	63	54	40	59	181	555	1	136,080
17	90	233	88	0	0	978	5	1845,360	25	72	32	59	88	203	598	0	135,936
59	164	114	90	0	121	748	4	1682,640	40	43	48	64	124	183	1126	3	132,096
7	125	123	119	90	0	1001	3	1829,625	8	72	31	59	160	203	598	1	131,688
44	164	114	76	0	121	838	4	1420,896	6	29	64	66	204	163	914	3	122,496
16	101	112	125	90	123	1001	2	1414,000	31	38	46	65	191	187	1001	2	113,620
15	65	117	163	165	117	262	3	1239,615	28	30	69	54	60	112	1066	3	111,780
35	88	112	125	0	0	1066	5	1232,000	20	32	48	64	197	181	432	1	98,304
22	68	114	155	164	121	748	4	1201,560	55	27	67	54	60	112	1120	3	97,686
3	172	83	81	0	152	667	0	1156,356	2	40	53	44	192	163	544	1	93,280
37	121	133	64	101	0	598	1	1029,952	23	44	36	53	170	81	379	5	83,952
48	60	123	125	0	112	1066	4	922,500	36	25	38	83	191	187	1066	5	78,850
49	63	116	117	170	0	262	1	855,036	13	38	37	45	185	184	1149	3	63,270
56	125	54	123	0	181	432	5	830,250	43	36	44	38	197	181	496	4	60,192
26	89	114	64	0	121	914	4	649,344	45	24	31	72	59	181	595	5	53,568
42	64	114	80	88	0	1120	5	583,680	9	37	42	26	0	181	627	4	40,404
50	74	114	69	152	0	1120	2	582,084	10	37	38	20	90	163	973	2	28,120
12	117	70	65	101	133	598	5	532,350	41	36	44	15	0	117	417	0	23,760
29	114	69	59	90	163	914	2	464,094	33	26	24	37	125	205	544	5	23,088
									46	26	37	17	127	163	973	2	16,354

Best Generation in 599 with fitness value of **94.63%** and **57** boxes packed.

performed for 100 instances of the I-DBLF algorithm with strongly heterogeneous and weakly heterogeneous boxes. A total of twelve experiments were executed with three subgroups based on the constraints for box rotation. The data were generated randomly for the 100 instances in the experiment. All the experiments featured the following GA parameters with the maximum generation set at 1000 with a population size of 100 and chromosomes (boxes) ranging from 60 to 100. The crossover rate was set to 70% with a mutation rate of 40%. Tournament and roulette wheel selections were both considered, with the final results based on tournament selection due to its minor benefit (an improvement of less than 3%). The tournament size was set to six for the following 100 instances for the random generation of boxes.

Table 9 shows the final results for our 100 instances regarding the random generation of strongly heterogeneous boxes; Fig. 12 illustrates two charts generated for 60 and 100 boxes sorted in descending order by the utilization rate. Table 10 shows the final results for our 100 instances regarding the random generation of weakly heterogeneous boxes; Fig. 13 illustrates two charts generated for 60 and 100 boxes sorted in descending order by the utilization rate.

Table 9
Results for strongly heterogeneous boxes.

	60 Boxes no rotation	60 Boxes 2-way	60 Boxes 6-way	100 Boxes no rotation	100 Boxes 2-way	100 Boxes 6-way
Average	91.07	92.12	97.40	91.83	93.93	96.49
SD	3.52	3.73	0.82	1.87	1.31	0.58
Min	82.14	84.23	93.75	86.66	90.06	94.54
Max	96.26	97.50	98.17	94.9	96.44	97.40

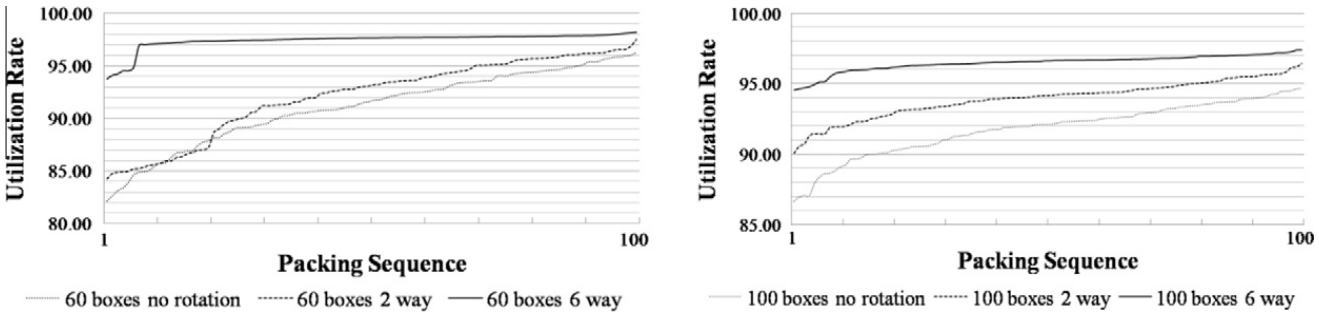

Fig. 12. Results graph for 60 and 100 strongly heterogeneous boxes.

Table 10
Results for weakly heterogeneous boxes.

	60 Boxes no rotation	60 Boxes 2-way	60 Boxes 6-way	100 Boxes no rotation	100 Boxes 2-way	100 Boxes 6-way
Average	85.35	89.56	92.39	87.68	90.26	91.50
SD	1.61	1.54	1.60	0.88	0.75	0.71
Min	81.62	84.60	87.58	85.49	88.64	88.48
Max	88.91	92.00	94.79	91.02	91.87	93.40

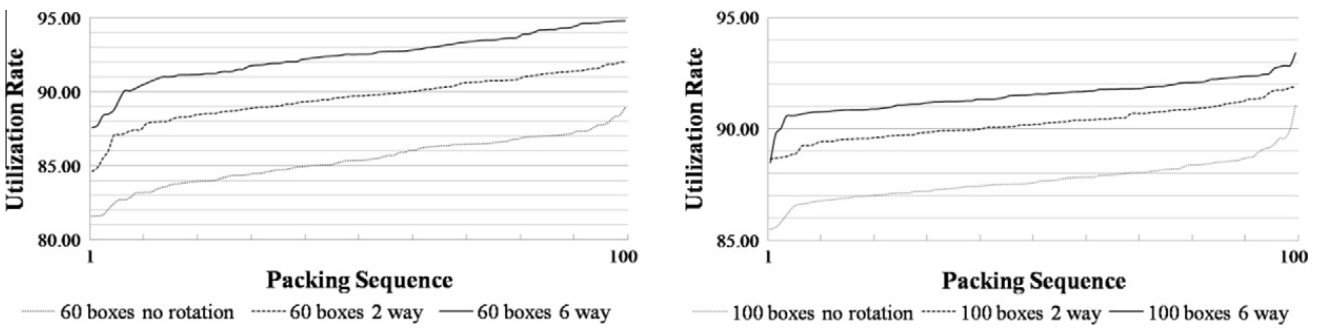

Fig. 13. Results graph for 60 and 100 weakly heterogeneous boxes.

7. Conclusions

This article presents the Improved Deepest Bottom Left with Fill (I-DBLF) algorithm, which utilizes a hybrid genetic algorithm to solve the 3D bin packing problem (3D-BPP). The I-DBLF packing strategy guarantees that a near-optimal solution for a set of boxes can be found in an acceptable amount of time. The rotation of boxes was considered to mimic practical application in container loading processes. The I-DBLF algorithm is based on three concepts. The first is to optimize the efficiency of the computational process by comparing the dimensions of cuboid space objects (the allowable packing area within the container) to those of boxes prior to packing processes. The second is to ensure that each box is packed at the deepest (smallest z value), bottom (smallest y value), left (smallest x value) position. The third is to find as many distinct packing sequences in the search space to populate the genetic algorithm.

In the future, we would like to consider variations of the 3D-BPP by introducing additional constraints related to the cargo value or priority, container weight limitations, and multiple containers.

Acknowledgement

The authors are grateful to the Editor-in-Chief and anonymous referees for their valuable comments. This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (**NRF**) funded by the Ministry of Education, Science and Technology (2010-0025714).

References

[1] K. Karabulut, M.M. Mustafa, A hybrid genetic algorithm for packing in 3D with deepest bottom left with fill method, ADVIS 2004: Advances in Information Systems, 2004, pp. 441–450.
[2] E.E. Bischoff, M.S.W Ratcliff, Issues in the development of approaches to container loading, Omega 23 (1995) 377–390.
[3] E.E. Bischoff, M.S.W. Ratcliff, Loading pallets with non-identical items. European Journal of Operational Research 84 (1995) 681–692.
[4] A. Lim, B. Rodrigues, Y. Wang, A multi-faced buildup algorithm for three-dimensional packing problems, Omega 31 (2003) 471–481.
[5] J. Hemminki, Container loading with variable strategies in each layer, Presented at ESI-X, EURO Summer Institute, Jouy-en-Josas, France, 1994, pp. 2–15.
[6] A.H. Bortfeldt, D.M. Gehring, A genetic algorithm for solving the container loading problem, International Transactions in Operational Research 4 (1997) 401–418.
[7] A.H. Bortfeldt, D.M. Gehring, A hybrid genetic algorithm for the container loading problem, European Journal of Operational Research 131 (2001) 143–161.
[8] A.H. Bortfeldt, D.M. Gehring, A parallel genetic algorithm for solving the container loading problem, International Transactions in Operational Research 9 (2002) 497–511.
[9] S.C. Liang, C.Y. Lee, S.W. Huang., A hybrid meta-heuristic for the container loading problem. Communications of the IIMA 7, no. 4, 2007, pp. 73–84.
[10] B.S. Baker, E.G. Coffman, R.L. Rivest, Orthogonal packing in two dimensions, SIAM Journal on Computing 9 (4) (1980) 846–855.
[11] E. Hopper, Two-dimensional packing utilising evolutionary algorithms and other metaheuristic methods. A Thesis submitted for the Degree of Doctor of Philosophy: London, University of Wales, Cardiff, 2000.
[12] E.E. Bischoff, Three dimensional packing of items with limited load bearing strength, European Journal of Operational Research 168 (2004) 952–966.
[13] A. Moura, J.F. Oliveira, A GRASP approach to the container loading problem, IEEE Intelligent Systems 20 (2005) 50–57.
[14] J.H. Holland, Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence, University of Michigan Press 8 (1975) 183.
[15] G.R. Raidl, G. Kodydek, Genetic Algorithms for the Multiple Container Packing Problem, Parallel Problem Solving from Nature – PPSN V: Lecture Notes in Computer Science 1498 (1998) 875–884.
[16] T. Fanslau, A.H. Bortfeldt, A tree search algorithm for solving the container loading problem, INFORMS Journal on Computing 22 (2) (2010) 222–235.
[17] S. Jakobs, Theory and methodology: on genetic algorithms for the packing of polygons, European Journal of Operational Research 88 (1996) 165–181.
[18] W. Zhu, Bottom Left Heuristics for 2D and 3D packing problems. http://rtsap.blogspot.com/2009_05_01_archive.html, May 2009.