Proceedings of the 2014 Industrial and Systems Engineering Research Conference
Y. Guan and H. Liao, eds.

A genetic algorithm for the three-dimensional bin packing problem with heterogeneous bins

Xueping Li, Zhaoxia Zhao, Kaike Zhang
Department of Industrial and Systems Engineering University of Tennessee, Knoxville
522 John D. Tickle Building 851 Neyland Drive Knoxville, TN, 37996

Abstract

The three-dimensional bin packing problem (3D-BPP) is to select one or more bins from a set of available bins to pack three dimensional, rectangular boxes such that the usage of the bin space is maximized. 3D-BPP finds wide applications in pharmaceutical industry, transportation and packaging system. In the traditional 3D-BPP, the holding bins are of identical size, while the problem considered in this paper addresses the case where bins are heterogeneous, i.e., varying in size. We present a genetic algorithm along with a novel heuristic packing procedure. The packing heuristic procedure converts box packing sequence and container loading sequence encoded in a chromosome into a compact packing solution. The genetic algorithm is used to evolve such sequences. The algorithm is first applied to 12 industrial instances and then tested on randomly generated instances. The results demonstrate that solutions with high quality can be found within reasonable time.

Keywords
Bin packing, Genetic algorithm, Three-dimensional, Empty Maximal Spaces

1. Introduction

The three-dimensional bin packing problem (3D-BPP) is a well known operations research optimization problem that directly models many industrial applications, i.e., container and pallet loading, TV program scheduling, cargo and warehouse management. The 3D-BPP considers to pack m rectangular boxes into unlimited identical rectangular containers (bins) so that the number of bins used is minimized, or equivalently, the bins' utilization ratio is maximized. We use containers and bins interchangeably. In this paper, we study a more general variant of 3D-BPP in which the bins used to pack items can be heterogeneous, that is, the size of bin varies from one to another and six-way orientations of box's are considered.

In shipping and transportation industry, several types of standard containers with different dimensions are used. For marine transport, standard containers of 20 or 40 feet length are used. For land or air transport, many kinds of containers are used [17]. Since the problem we consider is a generalization of original 3D-BPP, it suffers the computational difficulty that raised in 3D-BPP. In addition, our problem is more complicated because it need to determine which bins are selected for packing.

The 3D-BPP, which focus on packing rectangular items orthogonally into a minimal number of rectangular containers of identical size, is already NP-hard [14]. During last decades, both exact and heuristic algorithms for this problem have been developed. Martello et al. [14] present exact and approximation algorithms for 3D-BPP and discuss the lower bounds for the 3D-BPP. Fekete et al. [6] have shown that higher-dimensional packing problems of considerable size can be solved to optimality in reasonable time by making use of a structure and develop a two-level tree search algorithm for solving higher-dimensional packing problems to optimality. 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. Faroe et al. [5] provide a heuristic based on guided local search for packing items with fixed orientation into a minimum number of identical bins. Lodi et al. [12] have developed a tabu search framework by exploiting a new constructive procedure for the problem, and a unified tabu search code for general multi-dimensional bin packing problems is developed later by [13]. More recently, Gonçalves and Resende [7] present biased random-key genetic algorithm (BRKGA) for 2D and

3D bin packing problems in which a novel placement heuristic is proposed and hybridized in the BRKGA. They use a list of empty maximal-spaces to manage feasible placement positions.

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. Martello et al. [14] present a two-level Branch & Bound algorithm which gives optimal solution for packing items into a single bin. Kang et al. [9] 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. Another variation of the problem is referred to as open dimension problem (ODP), where a single variable dimension occurs in packing planning. Bortfeldt and Mack [1] present a layer-building heuristic method for the three-dimensional strip packing problem (3D-SPP) which is aim to packing items into single container to minimize required container length. Wu et al. [19] study a 3D-BPP with variable bin height. They first propose a exact mathematical model for based on Chen et al. [3]'s model and then designed a GA hybridize with a heuristic packing procedure. This problem is further studied by He et al. [8]. The authors develop an improved genetic algorithm(IGA) 3D-BPP with variable carton orientations.

Unlike the 3D-BPP with single or identical bins, not much attention has been paid in the literature so far to the problem with multiple heterogeneous bins which is also refereed as multiple container loading problem. Chen et al. [3] is among the first providing a 0-1 mixed integer linear programming (MILP) model to solve the 3D-BPP with variable orientations and various bins sizes. The model uses left, right, top and bottom decision variables to determine the relative positions of items and orientation variables to mimic different packing orientation. A similar problem is studied by Takahara and Miyamoto [17]. An evolutionary approach using GA is proposed, where a pair of sequences (one for boxes and one for containers) is used as the genotype, and a heuristic is used to determine the loading plan given the sequence of boxes and containers. They use a package loading heuristic called "Branch Heuristics" to convert a chromosome to a packing solution. A obvious deficient of this heuristic procedure is that it fails to make full use of available space. Eley [4] proposes a new approach based on a set partitioning formulation and use a tree search based heuristic to pre-generate single bin packing patterns. This model is extended by Che et al. [2] and Zhu et al. [20], who propose a new heuristic and a approximation algorithm to generate columns.

In this paper, we study a 3D-BPP with variable orientations and various bins sizes,which is NP-hard in the strong sense. A solution algorithm that hybridizes GA with a novel heuristic packing procedure is proposed. The heuristic packing procedure uses the concept of empty maximal spaces to manage free spaces in bins and select placement of boxes by match criterions. These sequences are converted to packing solutions by novel heuristic packing strategy based on concept of empty maximal space and evolve in the GA which leads to good solutions. Computational experiments are carried out on industrial instances and randomly generated instances. The computational results show that our GA can efficiently solve moderate size instances and give relative good solutions.

2. Genetic algorithm

The proposed algorithm is based on GA and a new heuristic packing strategy. The heuristic packing procedure uses the concept of empty maximal spaces to efficiently manage free space in containers and select placement combination of box and space by match criterions. The heuristic packing strategy generates packing solution based on a given box packing sequence (BPS) and a container loading sequence (CLS). The GA is used to evolve such sequences. Essential elements of GA consist the schemes of chromosome encodings, crossover, mutation, and selection. We design schemes for these essential elements to fit specific problem. Figure 1 illustrates the algorithm framework. We will discuss these key elements of our algorithm in this section respectively.

2.1 Chromosome encodings scheme

One chromosome has two parts consisting the box packing sequence (BPS) s^1 and container loading sequence (CLS) s^2. We use order based encoding scheme to present the sequences, therefore ,s^1 and s^2 is a permutation of $\{1,...,m\}$ and $\{1,...,N\}$ respectively, where m is the number of boxes and N is the number of containers.

2.2 Population Initialization

The initialization mechanism follows the one used in [18] to heuristically generate some special chromosomes in first generation. This mechanism is base on the observation that the bigger products should be packed into the box earlier. Gene sequence s^1 are generated through sorting descending according to the value of volume, length, width and height of each products respectively, while s^2 is generated randomly in their possible domain. Therefore, four

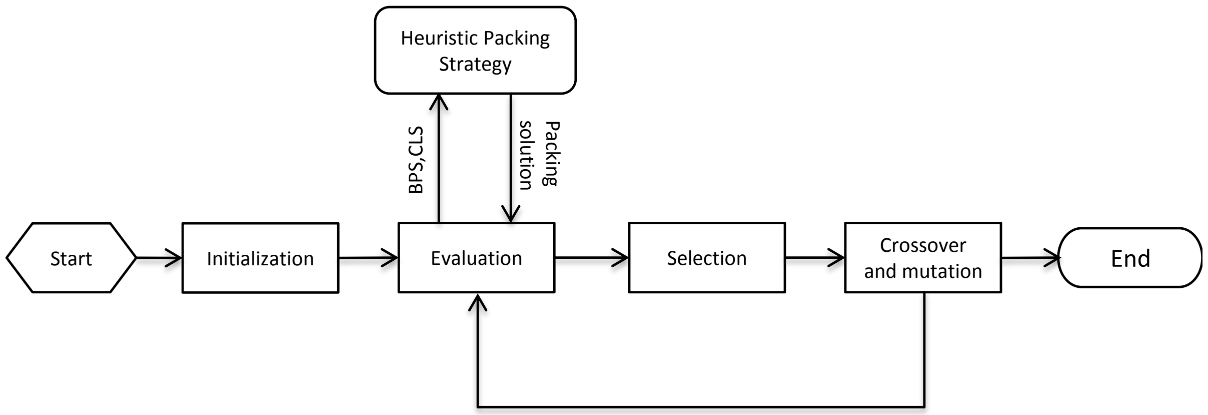


Figure 1: Genetic algorithm framework

special chromosomes are generated. The rest individuals in the first generation are generated randomly.

2.3 Selection

The tournament method for selection is adopted in our GA. A generation consists of pop chromosomes. They are sorted in descending fitness order. The fitness of a chromosome is given by a heuristic packing strategy which is described in section 3. First E chromosomes are selected as elitism which directly proceed into the next generation. $pop - E$ parents are selected into the mating pool using the tournament selection method with the tournament size equal to 2. In each round, we randomly select two different chromosomes from population. With probability $prob_t$ the better one(larger fitness) is added into the mating pool, otherwise the weak one is added. We select successive two chromosomes in the mating pool as parents. For each pair of parents, there is a probability $prob_c$ that they go directly into next generation, otherwise two offsprings are generated through crossover.

2.4 Crossover and mutation

Unlike binary encoding scheme that facilitates simple crossover operations like one-point crossover where gene strings are simply swapped with single cut point, designing crossover operation for order based encoding scheme is not such straight forward. Scholars have developed several crossover operators for this type of chromosomes, e.g.,partially matched crossover (PMX),cycle crossover (CX),order crossover operator (OX) etc. [16]. Two cutting points are randomly selected for the gene sequence, say i and j. Parents $P1$ and $P2$ will generate two offsprings $O1$ and $O2$. Child $O1$'s genes sequence s^1 are generated as follows, genes at positions from $i+1$ to j are copied from $P1$. Other positions in $O1$ are filled with missing genes start from $j+1$ circularly. The missing genes is obtained by sweeping $P2$ circularly start from $j+1$ and checking whether it has appeared in $O1$, if not, then fill current position in $O1$ with gene from $P2$. Child $O2$ can obtained by exchanging the roles of $P1$ and $P2$. The other genes sequence s^2 is operated independently in the same manner. See Figure 2.

Mutation is conducted on each newly generated offspring with probability pm. For each gene sequence, two positions are randomly selected and genes on these positions are swapped.

3. Best match heuristic packing strategy

A chromosome is evaluated by a heuristic packing strategy which maps a chromosome into a packing solution. Given a box packing sequence (BPS) and a list of container loading sequence (CLS), following heuristic packing strategy converts these sequences to a packing solution. Tranditional Deepest Bottom Left with fill(DBLF) heuristic and its

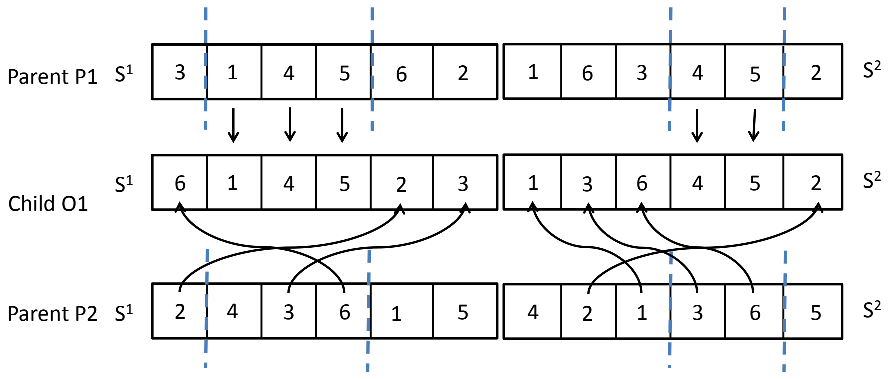


Figure 2: Illustration of crossover

variations are popular heuristics used for solving bin packing problems [10][18][9]. The heuristic always looks for space with minimal x (Deepest) coordinate to place current item, it uses z (Bottom) and y (Left) coordinates orderly as tie breakers. We notice this heuristic has two drawbacks: first, only one coordinate(x) plays dominant role in choosing candidate space; second, either the item or the space is first determined and then its counterpart is selected based on certain priority rules. Base on this observation, we propose what we call best match heuristic packing strategy. In the next sections we describe the main components of the placement strategy.

3.1 Empty maximal spaces

We use the concept of empty maximal spaces(EMSs) to represent the free spaces in bins, i.e. a list of largest empty cubic space available for packing which is not contained in any other space. Figure 3 illustrates the concept. Black frame donates the space currently available space for packing. When one blue box is placed in the corner, it generates three empty maximal spaces which are represented by yellow cubics. This concept is recently used in [7] and [15] to solve bin packing problems. The maximal spaces are represented by their vertexes with minimum and maximum coordinates, (x_i,y_i,z_i) and (X_i,Y_i,Z_i) respectively. We use the difference process(DP) introduced by [11] to update these empty maximal spaces. We follow the elimination rules proposed by [7] to accelerate the process.

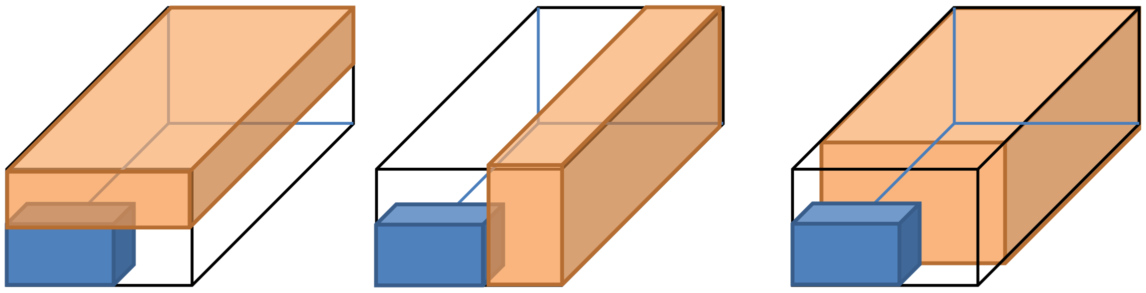


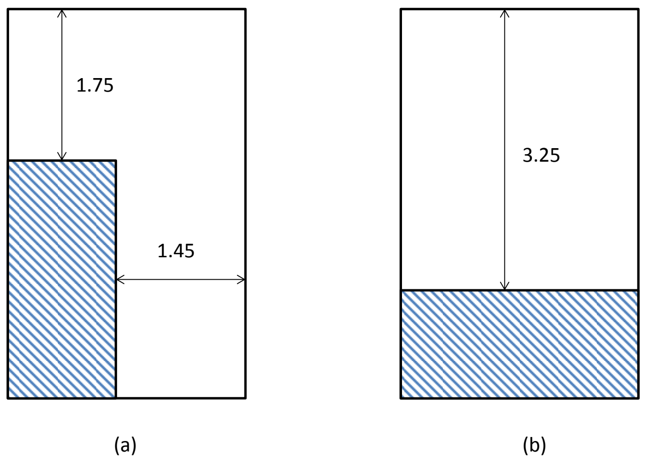
Figure 3: Empty maximal spaces

3.1.1 Priority of empty maximal spaces

We define priority of empty maximal spaces by following rules: for two empty maximal spaces with their minimum vertex coordinates (x_1,y_1,z_1) and (x_2,y_2,z_2). We first compare the smallest coordinates value of the two vertexes, the vertex with smaller one will give high priority, if they tie, we compare the second smallest values and sign high priority to the vertex with smaller one, if the two values are still the same, we compare the third smallest values, which is also the largest values. For example, to compare two vertexes $(3,5,4)$ and $(6,3,3)$, first we compare 3 and 3 which are the smallest values in each vertex coordinate, since they get the same value, we continue to compare the second smallest ones, that is 4 for the first vertex and 3 for the second one, since $4 > 3$, we sign high priority to the second vertex. The reason behind this prioritization is that we want to place boxes first in one corner and its adjacent sides of the container, then its inner space. We sort the empty spaces in each bin according to the priority defined above after we update of the EMSs each time.

3.2 Placement selection

The placement assignment is determined by applying a selection criteria to several possible placement candidates. In each iteration, first *kb* unpacked boxes are chosen based on the order given by BPS. Then first *ke* EMSs in the bin currently considered are selected with the order defined above. For these *kb* boxes and *ke* EMSs, we find all the feasible placement assignment with 6-way orientations of box. Then the (box , empty maximal space) pair with the largest fill ratio is first chosen for the placement. When one box has several feasible placements in one empty space, the one has smallest margin is selected. This can be done by calculating as follows: the dimensions of the space is $(X_1 - x_1, Y_1 - y_1, Z_1 - z_1)$ as defined, define the dimensions of the boxes after rotation as (l', w', h'), substract two vectors and we get $(X_1 - x_1 - l', Y_1 - y_1 - w', Z_1 - z_1 - h')$ which represents the margins to three faces of the space. Then, the priority of the placement is defined the same as the one in selecting empty maximal space. Figure 4 illustrates the idea in a two dimensions case.



The placement in (a) has a margin of (1.45,1.75) and (b) has a margin of (0,3.25). Our method will choose the placement in (b), since it generates less margin in one direction.

Figure 4: Orientation selection

If there is no feasible placement assignment for current boxes and spaces selection, we will try next *ke* EMSs in the container, the process continues until at least one feasible placement is find. If all EMSs in current container are tried, we move forward to try EMSs in the next opened container. If all opened containers are tried without finding one feasible placement, we open the next unopened container in order given by CLS. If there is no container in the list, the algorithm stops without finding a feasible packing solution. In this case, we set the fitness of the individual to be zero. Otherwise, we always find one or more feasible placement assignments and select one with largest fill ratio. Then we place the item in the chosen empty space, remove the item from the unpacked items list and update EMSs with the method proposed in Lai and Chan [11]. If there is no element in the unpacked boxes list, the algorithm terminates with solution found and the fitness of the individual is set to be the fill ratio. Algorithm 1 gives the pseudo-code of the packing heuristics.

Algorithm 1: : Best Match Heuristic Packing Procedure

Input: Boxes packing sequneces *BPS*, container loading sequence *CLS*

Output: A packing solution or **null** if not found

```
Let $OC$ be the list of opened containers
$OC \leftarrow \emptyset$
while $BPS \neq \emptyset$ do
    Let $P$ be a priority queue of candidate placements with the priority defined in section 3.2
    $boxplaced \leftarrow$ false
    for each opened container $c$ in $OC$ do
        Let EMSs be the empty maximal spaces in $c$
        $j \leftarrow 1$
        while $j \leq EMSs.size()$ and $boxplaced = false$ do
            $k \leftarrow j + ke$
            while $j < k$ and $j \leq EMSs.size()$ do
                for $i = 1$ to $kb$ and $i \leq BPS.size()$ do
                    for all 6 orientations $bo$ do
                        if Box $BPS_i$ can be placed in $EMSs_j$ with orientation $bo$ then
                            Add this placement combination to $P$
                $j \leftarrow j + 1$
            if $P \neq \emptyset$ then
                Make the placement indicted by $P_1$
                Update $EMSs$
                $boxplaced \leftarrow$ true
        if $boxplaced = true$ then
            break
    while $CLS \neq \emptyset$ and $boxplaced = false$ do
        Let EMS be the initial empty space in $CLS_1$
        $OC \leftarrow OC \cup CLS_1$
        $CLS \leftarrow CLS \setminus CLS_1$
        for $i = 1$ to $kb$ and $i \leq BPS.size()$ do
            for all 6 orientations $bo$ do
                if Box $BPS_i$ can be placed in EMS with orientation $bo$ then
                    Add this placement combination to $P$
        if $P \neq \emptyset$ then
            Make the placement indicted by $P_1$
            Update $EMSs$
            $boxplaced \leftarrow$ true
    if $boxplaced = false$ then
        return null
return Packing solution
```

4. Computational experiment

In this section, we summarize the computational results. The proposed algorithm is implemented in Java. All the experiments are carried out on a Dell OptiPlex 740 workstation with 3.1 GHz CPU and 2G memory running the Windows 7 operating system. The parameters used in GA are summarized in Table 1.

Table 1: Parameters used in the GA and the heuristic packing strategy

Generations	*pop*	*E*	$prob_t$	$prob_c$	*pm*	*kb*	*ke*
100	100	10	0.85	0.75	0.01	3	3

4.1 Results on industrial instances

The 12 instances are drawn from pharmaceutical industry. The data of the instances used in this study can be obtained on request. The instances are ranged from 12 boxes,8 containers to 78 boxes, 24 containers. We make a comparison of the performance of the proposed algorithm with the several other solution methods. Table 2 summaries the computational results. The column “#Items” and “#Bins” indicate the size of the instance with the number of items and the number of bins. Column “Time” reports the the CPU time in seconds used for solving the instance. The column “WS%” reports the percentage of wasted spaces in the solution. Because the goal is to minimize the percentage of wasted spaces, therefore the lower this value is, the better the solution is.

We first make a comparison of the MILP model proposed by [3] which is solved by CPLEX 12.1 MIP solver. The results are showed in column "CPLEX". In the column, the “Time” also reports the time limitation for the solver if optimal solution cannot be found in the limited time. For example, “6/100” means that a feasible solution is found within 6 seconds, however, it cannot be proved to optimal in the time limitation of 100 seconds. In order to test the efficiency of our heuristic packing strategy, we adapt different heuristic packing strategies from literature and hybridize them with the GA. The column “GA+CPDBLF” reports the results for hybridizing the GA with deepest bottom left heuristic [10] with using corner points [14] to manage free spaces, the orientation of box is determine by a genes sequence which follows the operation described in [18], if it doesn’t fit then all other orientations are tried. The column “GA+EMSDBLF” reports the results for hybridizing the GA with deepest bottom left fill heuristic. In this algorithm empty maximal spaces are used to manage free spaces, and the orientation is determined in the same manner above. The last column "GA+EMSBestMatch" reports the algorithm we proposed in this paper.

From the results, we observe that CPLEX can find optimal solution for small size instances,i.e., 12 products, 8 boxes. However, it fails to find optimal even feasible solution for moderate-sized instance. The GAs always find good solutions. Also, the proposed heuristic packing strategy has a better performance comparing to DBLF method. Another observation is that using empty maximal spaces to manage free spaces is more efficient and effective than using corner points method.

4.2 Results on randomly generated instances

Due to lack of shared test data for these kinds of studies, the random instance generator used by [14] is extended to fit our study and five classes of test data are generated. Following five type of random boxes are considered where $W = H = D = 100$:

- Type 1: l_j randomly distributed in $[1, \frac{1}{2}L]$, w_j randomly distributed in $[\frac{2}{3}W, W]$, h_j randomly distributed in $[\frac{2}{3}H, H]$
- Type 2: l_j randomly distributed in $[\frac{2}{3}L, L]$, w_j randomly distributed in $[1, \frac{1}{2}W]$, h_j randomly distributed in $[\frac{2}{3}H, H]$
- Type 3: l_j randomly distributed in $[\frac{2}{3}L, L]$, w_j randomly distributed in $[\frac{2}{3}W, W]$, h_j randomly distributed in $[1, \frac{1}{2}H]$
- Type 4: l_j randomly distributed in $[\frac{1}{2}L, L]$, w_j randomly distributed in $[\frac{1}{2}W, W]$, h_j randomly distributed in $[\frac{1}{2}H, H]$
- Type 5: l_j randomly distributed in $[1, \frac{1}{2}L]$, w_j randomly distributed in $[1, \frac{1}{2}W]$, h_j randomly distributed in $[1, \frac{1}{2}H]$

For each class $k(k = 1, ..., 5)$, boxes of type k are chosen with probability of 60%, and the rest four types are chosen with probability 10% each. For each class, we consider instances with a number of boxes equal to 20,50,100,150 and a

Table 2: Algorithms performance comparison on 12 industrial instances

#Inst	#Items	#Bins	CPLEX		GA+CPDBLF		GA+EMSDBLF		GA+EMSBestMatch	
			Time	WS%	Time	WS%	Time	WS%	Time	WS%
1	12	8	16s	26.67%	6.6	26.67%	3.7	26.67%	6.6	26.67%
2	12	8	6s/100s	33.25%	6.7	33.25%	1.6	33.25%	4.2	33.25%
3	12	8	3s	50.64%	5.7	50.64%	2.9	50.64%	7.9	50.64%
4	20	8	31s/180s	42.17%	17.3	22.47%	4.4	22.47%	9.8	18.30%
5	20	8	170s/180s	37.59%	17.3	35.00%	3.5	23.18%	7.1	21.34%
6	20	8	25s/180s	37.02%	18.7	22.48%	4.3	22.48%	9.8	20.62%
7	50	10	840s/1600s	52.91%	158.9	30.72%	24.7	29.59%	62.7	22.48%
8	50	10	-	-	153.5	33.39%	34.8	23.35%	35.5	23.35%
9	50	10	-	-	149.5	28.51%	31.1	31.98%	43.2	20.82%
10	78	15	-	-	319	37.31%	53.2	26.76%	105.5	22.05%
11	78	15	-	-	376	38.41%	55.6	27.43%	106.9	22.76%
12	78	24	-	-	364.5	36.70%	59.9	25.26%	63.5	23.40%

- No feasible solution is found in 3600 seconds.

Table 3: Algorithm performance on random instances

Class	#Item	#Bins	Time used			Percent of wasted			#Bins used
			Min	Average	Max	Min	Average	Max	
Class1	20	10	4	7	14	18.03%	21.40%	27.86%	2.4
	50	20	14	28	70	17.50%	20.19%	23.55%	5.6
	100	40	42	93	198	13.46%	18.11%	21.78%	9.7
	150	50	93	132	218	12.89%	17.09%	21.11%	15.9
Class2	20	10	4	7	9	13.43%	20.00%	26.04%	2.3
	50	20	21	44	79	17.66%	18.70%	20.97%	4.5
	100	40	54	100	252	15.74%	17.99%	20.33%	9.5
	150	50	87	195	331	14.80%	17.92%	21.16%	13.3
Class3	20	10	4	7	15	17.30%	22.91%	27.24%	2.6
	50	20	20	44	109	17.01%	19.74%	22.52%	4.7
	100	40	73	114	226	16.88%	18.19%	20.19%	8.9
	150	50	82	126	179	12.57%	16.78%	20.04%	15.3
Class4	20	10	3	5	9	16.92%	22.66%	32.55%	3.4
	50	20	9	16	37	19.77%	22.20%	23.90%	9.1
	100	40	26	39	76	15.52%	19.31%	23.75%	18.2
	150	50	45	68	95	12.66%	17.59%	23.88%	28.6
Class5	20	10	4	8	10	15.47%	22.36%	29.50%	2
	50	20	47	71	173	14.01%	19.69%	25.10%	2.7
	100	40	100	245	525	14.42%	17.45%	21.77%	5.1
	150	50	185	283	396	9.05%	13.64%	17.12%	7.7

corresponding number of available containers 10,20,40,50. All containers are generated with each of their dimensions randomly distributed in [50,200]. For each class and an instance size, we generate 10 random instances. The results are showed in Table 3. For most instances, the algorithm ends within acceptable amount of time and the solution time increase moderate with the increase of instance size. Most difficult instances come from class 5 while they have a better utilization ratio with only 13.64% space is wasted on average.

5. Conclusion

In this paper, we study a variant of the three dimensional bin packing problem (3D-BPP) which consists in packing, with no overlapping, a set of three-dimensional rectangular shaped products into the a number of three-dimensional rectangular shaped bins with various bin sizes to maximize the space utilization ratio. A GA hybridized with a novel heuristic packing strategy is proposed. Computational experiments are carried out over a set of test industrial instances to examine the performance of the proposed algorithm. The computational result demonstrates that the proposed GA is able to give good solution to moderate size problem in reasonable time.

References

[1] Bortfeldt, A., Mack, D., 2007. A heuristic for the three-dimensional strip packing problem. European Journal of Operational Research 183 (3), 1267–1279.

[2] Che, C. H., Huang, W., Lim, A., Zhu, W., 2011. The multiple container loading cost minimization problem. European Journal of Operational Research 214 (3), 501–511.

[3] Chen, C., Lee, S., Shen, Q., 1995. An analytical model for the container loading problem. European Journal of Operational Research 80 (1), 68–76.

[4] Eley, M., 2003. A bottleneck assignment approach to the multiple container loading problem. OR Spectrum 25 (1), 45–60.

[5] Faroe, O., Pisinger, D., Zachariasen, M., 2003. Guided local search for the three-dimensional bin-packing problem. INFORMS Journal on Computing 15 (3), 267–283.

[6] Fekete, S. P., Schepers, J., van der Veen, J. C., 2007. An exact algorithm for higher-dimensional orthogonal packing. Operations Research 55 (3), 569–587.

[7] Gonçalves, J. F., Resende, M. G., 2013. A biased random key genetic algorithm for 2d and 3d bin packing problems. International Journal of Production Economics 145 (2), 500–510.

[8] He, Y., Wu, Y., de Souza, R., 2012. A global search framework for practical three-dimensional packing with variable carton orientations. Computers & Operations Research 39 (10), 2395–2414.

[9] Kang, K., Moon, I., Wang, H., 2012. A hybrid genetic algorithm with a new packing strategy for the three-dimensional bin packing problem. Applied Mathematics and Computation 219 (3), 1287–1299.

[10] Karabulut, K., İnceoğlu, M. M., 2005. A hybrid genetic algorithm for packing in 3d with deepest bottom left with fill method. In: Advances in Information Systems. Springer, pp. 441–450.

[11] Lai, K., Chan, J. W., 1997. Developing a simulated annealing algorithm for the cutting stock problem. Computers & industrial engineering 32 (1), 115–127.

[12] Lodi, A., Martello, S., Vigo, D., 2002. Heuristic algorithms for the three-dimensional bin packing problem. European Journal of Operational Research 141 (2), 410–420.

[13] Lodi, A., Martello, S., Vigo, D., 2004. Tspack: a unified tabu search code for multi-dimensional bin packing problems. Annals of Operations Research 131 (1-4), 203–213.

[14] Martello, S., Pisinger, D., Vigo, D., 2000. The three-dimensional bin packing problem. Operations Research 48 (2), 256–267.

[15] Parreño, F., Alvarez-Valdes, R., Tamarit, J. M., Oliveira, J. F., 2008. A maximal-space algorithm for the container loading problem. INFORMS Journal on Computing 20 (3), 412–422.

[16] Sivanandam, S., Deepa, S., 2007. Introduction to genetic algorithms. Springer.

[17] Takahara, S., Miyamoto, S., 2005. An evolutionary approach for the multiple container loading problem. In: HIS 05: Proceedings of the Fifth International Conference on Hybrid Intelligent Systems. IEEE, pp. 227–232.

[18] Wang, H., Chen, Y., 2010. A hybrid genetic algorithm for 3d bin packing problems. In: Bio-Inspired Computing: Theories and Applications (BIC-TA), 2010 IEEE Fifth International Conference on. IEEE, pp. 703–707.

[19] Wu, Y., Li, W., Goh, M., de Souza, R., 2010. Three-dimensional bin packing problem with variable bin height. European Journal of Operational Research 202 (2), 347–355.

[20] Zhu, W., Huang, W., Lim, A., 2012. A prototype column generation strategy for the multiple container loading problem. European Journal of Operational Research 223 (1), 27–39.