Three Dimensional Bin Packing Problem with Variable Bin Height

Yong Wu [a,b,*], Wenkai Li [b], Mark Goh [b,c], Robert de Souza [b]

[a]*Institute for Logistics and Supply Chain Management, Victoria University, PO Box 14428, VIC 8001 Australia*

[b]*The Logistics Institute – Asia Pacific, National University of Singapore, Block E3A Level 3, 7 Engineering Drive 1, Singapore 117574.*

[c]*NUS Business School, National University of Singapore, 1 Business Link, Singapore 117592*

Abstract

This paper studies a variant of the three-dimensional bin packing problem (3D-BPP), where the bin height can be adjusted to the cartons it packs. The bins and cartons to be packed are assumed rectangular in shape. The cartons are allowed to be rotated into any one of the six positions that keep the carton edges parallel to the bin edges. This greatly increases the difficulty of finding a good solution since the search space expands significantly comparing to the 3D-BPP where the cartons have fixed orientations. A mathematical (mixed integer programming) approach is modified based on Chen et al. (1995) and numerical experiments indicate that the mathematical approach is not suitable for the variable bin height 3D-BPP. A special bin packing algorithm based on packing index is designed to utilize the special problem feature and is used as a building block for a genetic algorithm designed for the 3D-BPP. The paper also investigates the situation where more than one type of bin are used and provides a heuristic for packing a batch of cartons using the genetic algorithm. Numerical experiments show that our proposed method yields quick and satisfactory results when benchmarked against the actual packing practice and the MIP model with the latest version of CPLEX.

Key words: bin packing, mixed integer programming, optimization, genetic algorithms, batch bin packing, three-dimensional

* Corresponding author. Phone: +61 3 9919 6272, fax: +61 3 9689 4859
Email addresses: `wuyong@pmail.ntu.edu.sg` (Yong Wu), `tlilw@nus.edu.sg` (Wenkai Li), `tligohkh@nus.edu.sg` (Mark Goh), `tlirbrtm@nus.edu.sg` (Robert de Souza).

1 Introduction

Packing cartons into a bin is an important material handling activity in the manufacturing and distribution industries. A bin is defined in this paper as a rectangular box used to hold smaller cartons. The cartons can be any rectangular stackable objects with different dimensions which can hold all kinds of goods and products for easy handling.

We assume that all cartons can be freely rotated and placed into the bin at one of the six positions that keep the carton edges parallel to the bin edges. It is also assumed, without loss of generality, that all the carton dimension data are positive values satisfying the only constraint that each carton can be placed into the bin in at least one of the six positions. No further restriction is present: the cartons need not to be packed in layers, and we do not impose the so-called guillotine constraint, which requires that the patterns be such that the items can be obtained by sequential face-to-face cuts parallel to the faces of the bin.

This paper investigates a unique variant of three-dimensional bin packing problem (3D-BPP). There are more than one kind of carton to be used and the bin height can be trimmed to fit the contents of the bin. Therefore, the actual height of each bin varies. The variable size bin packing problem (VS-BPP) contains the classical 3D-BPP, where all the bins have the same capacity and cost, as a particular case. The 3D-BPP is an NP-hard problem in a strong sense (Martello et al., 2000). The flexibility in bin height poses a greater challenge in providing quality solutions in a sense that unlike the traditional 3D-BPP where only the number of bins used needs to be optimized, the VS-BPP needs to search for the minimum overall volume for all bins. The flexibility of carton orientation also expands the search space significantly and hence increases the difficulty of finding optimal solutions.

The rest of the paper is structured as follows: The relevant literature is given in Section 2. Section 3 presents the mathematical model which aims to solve the single bin packing problem and some CPLEX results are reported. Sections 4 and 5 introduce the genetic algorithm for single and multiple bin packing problems, respectively. Section 6 concludes the paper.

2 Literature Review

Chen et al. (1995) provide a mixed integer programming formulation to solve the 3D-BPP without orientation restriction. Due to the strong NP-hardness of the problem, the MIP model can only solve instances where a few cartons to be packed to optimality. Recently, Puchinger and Raidl (2007) consider the two-dimensional BPP, where a new integer linear programming formulation is proposed and solved using CPLEX.

Most research focus on the BPP where identical bins are used and all the cartons have fixed orientation. For instance, Martello et al. (2000) and den Boef et al. (2005) present an exact algorithm for filling a single bin and discussed the lower bounds for the 3D-BPP. Martello et al. (2007) present an algorithm to solve moderately large instances to optimality for the general 3D-BPP and its robot-packable variant.

Due to the strong NP-hardness of the 3D-BPP, heuristics are often used and the carton orientation constraint normally can be relaxed. George and Robinson (1980) present a wall-building approach heuristic without restrictions on the carton orientation for identical containers. Bischoff et al. (1995) propose a pallet loading heuristic for non-identical cartons where the stability of the pallet is considered. Pisinger (2002) offers a heuristic based on the wall-building approach, where strips and layers are created so that the 3D-BPP can be decomposed into smaller sub-problems. Baltacioğlu et al. (2006) develop a new heuristic algorithm using rules to mimic human intelligence to solve the 3D-BPP. Bischoff and Marriott (1990) evaluate a number of heuristic variants for determining the efficient packing patterns in loading freight containers with rectangular items. George (1996) investigates the case of multiple container loading using the pipe packing as a case study. Faroe et al. (2003) provide a heuristic for packing cartons into a minimum number of identical containers based on guided local search, however, no carton rotation is allowed.

Genetic algorithms (Goldberg, 1989; Michalewicz, 1994) and tabu search are often used as the building blocks for heuristics to solve the 3D-BPP. For instance, Hopper and Turton (2001) investigate different heuristics for two dimensional packing problem; Puchinger et al. (2004) use genetic algorithms to solve a two dimensional glass cutting problem; Zhang et al. (2007) propose a heuristic algorithm using tabu search in combination with a bin-loading algorithm to solve the generic 3D-BPP; Gehring and Bortfeldt (1997) present a genetic algorithm for loading strongly heterogeneous sets of cartons into a single container. More recently, Crainic et al. (2008) introduce a two-level tabu search where the first level aims to reduce the number of bins and the second optimizes the packing of the bins for cartons with fixed orientations. Egeblad and Pisinger (2009) examine two and three-dimensional knapsack packing problem using simulated annealing. Lodi et al. (2002) provide a tabu search framework by exploiting a constructive heuristic to evaluate the neighborhood where the carton orientation is fixed.

Although the techniques for the BPP with identical bins can be adapted to solve the VS-BPP in a trial and error manner, research specific to the VS-BPP is relatively scarce. Among them, Alves and de Carvalho (2007) study column generation for the one-dimensional VS-BPP; Correia et al. (2008) propose a discretized model formulation based on a straightforward integer programming formulation solving optimally for the one-dimensional VS-BPP.

3 Single bin packing: Mathematical approach

3.1 Formulation

Chen et al. (1995) provide an analytical model to formulate the general container loading problem, where multiple containers can be used, into a 0-1 MIP model. We adopt the approach in Chen et al. (1995) and modify the mathematical model into a single bin packing problem with variable bin height. Although the multiple bin VS-BPP can be formulated similarly, we skip this model considering the fact that even the single bin VS-BPP problem is already NP-Hard.

The following variables are introduced for the mathematical formulation.

(l_i, w_i, h_i): parameters indicating the length, width, and height of carton i.

$(L, W, \tilde{H})$: length, width and height of the bin to be loaded, where $\tilde{H}$ indicates that the bin height can be adjusted.

(x_i, y_i, z_i): continuous variables for coordinates of carton i's left-bottom-behind corner.

$X_{l_i}, Z_{l_i}, Y_{w_i}, Z_{h_i}$: binary variables indicating whether the length direction of carton i is parallel to the bin's X and Z axes, the width direction is parallel to the Y axis, or the height direction is parallel to the Z axis, respectively, to determine the orientation of carton i.

a_{ij}, b_{ij}, c_{ij}: binary variables defining the relative placement of carton i to carton j: variables will be 1 if carton i is in front of, to the right of, or on top of carton j, respectively; otherwise, 0.

M: a large enough number.

The objective is to minimize the variable bin height $\tilde{H}$, i.e.

$$\min \tilde{H}$$

The constraints are as follows:

$$\begin{aligned} x_i + l_i X_{l_i} + w_i(Z_{l_i} - Y_{w_i} + Z_{h_i}) + h_i(1 - X_{l_i} - Z_{l_i} + Y_{w_i} - Z_{h_i}) \\ \leqslant x_j + M(1 - a_{ij}), \quad i \neq j \end{aligned} \tag{1a}$$

$$\begin{aligned} y_i + w_i Y_{w_i} + l_i(1 - X_{l_i} - Z_{l_i}) + h_i(X_{l_i} + Z_{l_i} - Y_{w_i}) \\ \leqslant y_j + M(1 - b_{ij}), \quad i \neq j \end{aligned} \tag{1b}$$

$$z_i + h_i Z_{h_i} + w_i(1 - Z_{l_i} - Z_{h_i}) + l_i Z_{l_i} \leqslant z_j + M(1 - c_{ij}), \quad i \neq j \tag{1c}$$

$$x_i + l_i X_{l_i} + w_i(Z_{l_i} - Y_{w_i} + Z_{h_i}) + h_i(1 - X_{l_i} - Z_{l_i} + Y_{w_i} - Z_{h_i}) \leqslant L \quad (2a)$$
$$y_i + w_i Y_{w_i} + l_i(1 - X_{l_i} - Z_{l_i}) + h_i(X_{l_i} + Z_{l_i} - Y_{w_i}) \leqslant W \quad (2b)$$
$$z_i + h_i Z_{h_i} + w_i(1 - Z_{l_i} - Z_{h_i}) + l_i Z_{l_i} \leqslant \tilde{H}; \quad (2c)$$

$$a_{ij} + a_{ji} + b_{ij} + b_{ji} + c_{ij} + c_{ji} \geqslant 1, \quad i \neq j \quad (3)$$

$$X_{l_i} + Z_{l_i} \leqslant 1 \quad (4a)$$
$$Z_{l_i} + Z_{h_i} \leqslant 1 \quad (4b)$$
$$Z_{l_i} - Y_{w_i} + Z_{h_i} \leqslant 1 \quad (4c)$$
$$Z_{l_i} - Y_{w_i} + Z_{h_i} \geqslant 0 \quad (4d)$$
$$1 - X_{l_i} - Z_{l_i} + Y_{w_i} - Z_{h_i} \leqslant 1 \quad (4e)$$
$$1 - X_{l_i} - Z_{l_i} + Y_{w_i} - Z_{h_i} \geqslant 0 \quad (4f)$$
$$X_{l_i} + Z_{l_i} - Y_{w_i} \leqslant 1 \quad (4g)$$
$$X_{l_i} + Z_{l_i} - Y_{w_i} \geqslant 0 \quad (4h)$$

Constraints (1) ensure that any two cartons i and j do not overlap with each other. Constraints (2) keep all cartons within the bin dimension. X_{l_i}, Z_{l_i}, Y_{w_i} and Z_{h_i} are used to calculate the respective mappings of carton length, width and height to the corresponding bin's X, Y and Z axes. Constraint (3) limits the relative position of any two cartons i and j. Constraints (4) ensure that the binary variables which determine the carton position are properly controlled to reflect practical positions.

3.2 MIP results

The above mathematical model was coded in GAMS using CPLEX 11.0, running on a 2.2 GHz Dual Core Pentium processor with 3GB RAM. The CPLEX was allowed to run for two hours, and the best bin height available was reported as the results of the mathematical model. The status of each run was also reported, i.e., proven optimal solution or terminated with time limit.

There are two types of bins used: small and big. The small bin has a gross planar dimension of 82×60 (unit length is centimeter), with usable dimensions of 78×57; the big bin has a gross planar dimension of 120×100, with usable dimensions of 116.5×97. The difference between the gross and usable dimension is for shrink

Table 1
Best available solution obtained by CPLEX in 2 hours for small bin cases

Case	Num of Ctn	Actual	MIP	Imprv (%)	Sol. Found T (s)	Status
SM01	8	91	80	12.09	153	Proven optimal
SM02	8	56	37	33.93	547	Proven optimal
SM03	9	69	47	31.88	1	Proven optimal
SM04	9	91	82	9.89	899	Time out
SM05	10	56	52	7.14	734	Proven optimal
SM06	12	53	38	28.30	285	Proven optimal
SM07	17	70	62	11.43	72	Time out
SM08	17	91	71	21.98	7,114	Time out
SM09	19	71	62	12.68	104	Time out
SM10	21	85	62	27.06	6,974	Time out

wrapping. Both small and big bins can pack up to 95cm high while in actual operation the height is trimmed to fit the contents of the bin. There is an additional height of 15cm for both a cushion layer (to protect the cartons) and a ‘pallet leg’ (to load/unload the bins by forklifts). Therefore, the maximum gross height for bins is 110cm. The cushion layer and the ‘pallet leg’ need not to be considered for single bin packing; but for multiple bins, they need to be taken into account since with proper choice of bins, the volume they occupy can be minimized. Ten test cases of the small bin and 27 cases of the big bin were collected from the actual operation of a large company where the operators manually decide on how cartons are placed in the bin based on a rule of thumb and their own packing experience. The test cases are moderate heterogeneous: there are 10 types of cartons to be packed, and occasionally there are cartons with customized sizes used. The biggest carton has a dimension of $63.1 \times 56.4 \times 46.4$ and the smallest carton has a dimension of $33 \times 21 \times 4$.

Tables 1 and 2 report the number of cartons packed in each bin, the actual heights from the actual operations, the results of the MIP model, the improvement of the MIP model against the actual operations, the time at which the best solution is found, and solver status for small bin and big bin cases, respectively. The actual heights were taken from the company for benchmarking purposes. All test cases are sorted in ascending order of the number of cartons to be packed in each table. It can be observed that CPLEX cannot find optimal solution except for a few cases within the time limit given for its runs. For cases with more than 12 cartons to be packed, CPLEX reports ‘time out’. However, CPLEX is able to provide better solutions for all small bin cases and most of the big bin cases except BM18, BM26 and BM27, where more than 30 cartons need to be packed. For certain cases, the improvement is quite significant. As the number of cartons to be packed goes up,

Table 2
Best available solution obtained by CPLEX in 2 hours for big bin cases

Case	Num of Ctn	Actual	MIP	Imprv (%)	Sol. Found T (s)	Status
BM01	5	91	84.1	7.58	3	Proven optimal
BM02	11	91	83.1	8.68	146	Time out
BM03	12	45	32	28.89	368	Proven optimal
BM04	13	91	77.4	14.95	4,406	Time out
BM05	15	91	79	13.19	6,490	Time out
BM06	16	67	40	40.30	168	Time out
BM07	21	48	40	16.67	377	Time out
BM08	22	59	53	10.17	1,588	Time out
BM09	23	94	82	12.77	485	Time out
BM10	23	91	63.1	30.66	1,059	Time out
BM11	25	91	78	14.29	1,190	Time out
BM12	26	94	68	27.66	167	Time out
BM13	27	83	78	6.02	6,490	Time out
BM14	28	95	88.4	6.95	5,648	Time out
BM15	29	91	87	4.40	6,516	Time out
BM16	29	63	52	17.46	6,697	Time out
BM17	29	91	88.4	2.86	7,160	Time out
BM18	30	70	73.4	−4.86	6,509	Time out
BM19	30	94	79	15.96	1,313	Time out
BM20	30	94	87	7.45	6,660	Time out
BM21	30	95	87	8.42	6,551	Time out
BM22	31	91	87	4.40	6,632	Time out
BM23	33	95	93.4	1.68	6,094	Time out
BM24	35	91	89	2.20	7,140	Time out
BM25	36	94	79	15.96	6,880	Time out
BM26	38	91	97	−6.59	7,110	Time out
BM27	40	91	109	−19.78	6,528	Time out

the time required to find good solutions normally increases.

The numerical experiments indicate that for bin packing problem in the company, CPLEX can be used for non-real-time solutions. However, CPLEX needs significant amount of computational time to generate a reasonably good result and this makes it unsuitable for guiding on site operations. Hence, heuristics are needed if quality and fast solutions are required.

4 Single bin packing: Genetic algorithms approach

Genetic algorithms (GA) are now widely applied in many fields such as computer science, bio-informatics, engineering, economics and operations research due to its easy implementation and good quality in providing solutions to difficult problems. We propose a GA for the VS-BPP and design a special routine for GA fitness evaluation, i.e., single bin packing based on packing index.

4.1 Representation

The GA chromosome needs to consider two factors: the order of cartons to be packed and the carton rotation positions. All cartons are first sorted in volumetric descending order and numbered. The rotation positions are represented by numbers from 1 to 6 to indicate the six positions a carton can be placed. A sequence of pairs of carton numbers and carton positions forms a GA chromosome.

4.2 Initial population

The initial population is generated by fixing the volumetric order of cartons while randomly assign rotation positions to cartons for each chromosome. Since the cartons are sorted in terms of their volume, we have $V_{c_1} \geqslant V_{c_2} \cdots \geqslant V_{c_n}$, where V_{c_i} is the volume of carton i. Two possible initial chromosomes could be:

1’2   2’3   3’1   4’5   5’2 $\cdots$

1’5   2’1   3’3   4’2   5’6 $\cdots$

The first half of each gene represents the carton number while the second half shows the rotation position. This is to utilize the feature of the first fit decreasing bin-packing heuristic algorithm to ensure good initial solutions can be generated (Dósa, 2007).

4.3 Reproduction

The reproduction is based on roulette wheel selection, which enables better chromosomes have higher chance to generate offsprings. We also applies the niche mechanism during reproduction: the best solution found so far is copied into the reproduction pool and replace 10% of the the total chromosomes. This not only ensures that the fitter genes stay in the evolutionary process, but also provides a chance to exploit the potential out of the best solution found during the evolution.

4.4 Crossover

The crossover is one-point crossover, a random location is selected for two parents and the two parts after the crossover point of the two parents are switched over to form two children. In this way, the order of cartons to be packed stays relatively stable since the order plays an important role in bin packing. If the newly formed children are not feasible due to some cartons appear in the children twice while some do not appear at all, the children need to be repaired.

Two repairing schemes are proposed: sequential repair and random repair. Sequential repair first identifies the missing bits and the duplicated bits in a chromosome and sorts them according to their carton numbers. Then it replaces the first duplicated bit with the first missing bit, the second duplicated bit with the second missing bit until all duplicated bits are replaced by the missing bits. Random repair creates one duplicated bit pool and one missing bit pool and replace the duplicated bit randomly within the missing bit pool. Preliminary results indicate that the random repair outperforms the sequential repair and is therefore used in the subsequent numerical tests.

4.5 Mutation

There are two kinds of mutation here: sequence mutation and position mutation. Sequence mutation selects two bits in a chromosome and switch them to form a new one. For position mutation, several bits are picked in the chromosome and are rotated into new random positions.

4.6 Fitness evaluation

Given a chromosome where the sequence of cartons has been determined and the carton orientations have been fixed, Algorithm 1 is used to evaluate the fitness

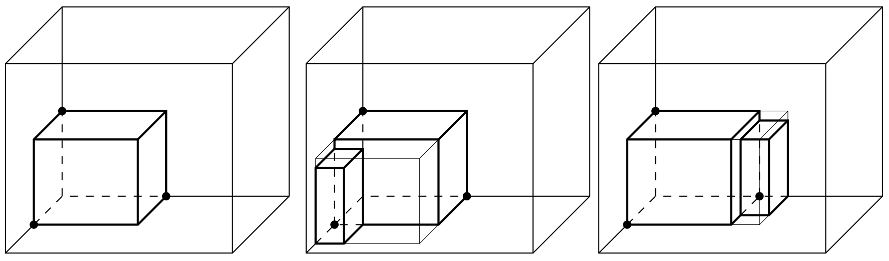
(a) Extreme points (b) Not preferred (c) Preferred

Fig. 1. Extreme points and the packing position selection

of the chromosome (to calculate the ultimate bin height). The algorithm is based on the concept of extreme points which are used in the literature (Crainic et al., 2008). After a carton has been placed into position, the left-front-bottom, right-behind-bottom and left-behind-top points are treated as extreme points as shown as bold dots in Figure 1(a). All available extreme points are collectively denoted as set P. The next carton i to be packed will try all available extreme points in set P and calculate an index for each point. For extreme point P_j, let $L_{P_j}, W_{P_j}, H_{P_j}$ denote the length, width and height of the convex hull after the carton i is placed at the extreme point P_j, let V_1^i denote the total volume of all cartons packed so far inclusive of carton i, the corresponding packing index I_{P_j} is calculated as:

$$I_{P_j} = \frac{L_{P_j} \times W_{P_j} \times H_{P_j}^2}{V_1^i}, \tag{5}$$

which actually is a measure of height over the fill rate of current convex hull. The extreme point with the highest index is selected for the packing point. If the carton i cannot be placed at an extreme point, the corresponding index is set to 0. Figures 1(b) and 1(c) show an illustration of the preferred extreme point selection since Figure 1(c) results in a more 'compact' packing.

Algorithm 1 presents the procedure for the GA fitness evaluation. It starts from the only extreme point of $(0, 0, 0)$. At the beginning of each iteration, the current height H_{i-1} of the convex hull of packed cartons is recorded. This is just a precaution for the situation when no existing extreme point can be used, the point $(0, 0, H_{i-1})$ will be used instead to ensure a feasible packing. The packing index is then calculated for each extreme point according to (5) and the point with the highest index is selected. The carton i is then moved along X-axis, Y-axis and Z-axis respectively to check whether the existing space between the cartons can be squeezed. The new location is then denoted as the point to put carton i and the corresponding extreme points are added into the extreme point set P. The whole process repeats until all cartons are packed.

Algorithm 1 Fitness evaluation

```
1:  $P = \{(0,0,0)\}$;
2:  for $i = 1$ to $n$ do
3:      Get the height $H_{i-1}$ of convex hull of currently packed cartons;
4:      for $P_j \in P$ do
5:          Calculate the index $I_{P_j}$;
6:      end for
7:      if no extreme point can be selected then
8:          $P = P \cup \{(0,0,H_{i-1})\}$;
9:          Calculate the index $I_{P_{(0,0,H_{i-1})}}$;
10:     end if
11:     Select the extreme point $P_j$ with highest index;
12:     Put carton $i$ at $P_j$ and delete $P_j$ from set $P$;
13:     Squeeze the carton along $X$-axis, $Y$-axis and $Z$-axis as much as possible,
        denote the new location as $\{(x_i, y_i, z_i)\}$;
14:     Put carton $i$ at $\{(x_i, y_i, z_i)\}$;
15:     Add the left-front-bottom, right-behind-bottom and left-behind-top points of
        carton $i$ as extreme points into set $P$;
16: end for
```

4.7 Experiments

The 37 cases used for the MIP model were tested using the GA and the test results are reported in Tables 3 and 4. The population size for small bin test cases is 120 and the code runs for 200 generations. For big bin test cases, the population size and number of generations are 200 and 500 respectively. For crossover and mutation rates, we evaluated combinations of crossover rate with 0.5, 0.7 and 0.9 and mutation rate with 0.1, 0.2, 0.3, 0.6 and 0.9. The results indicate that a higher crossover rate and a higher mutation rate perform better and therefore both rates are fixed at 0.9. Twenty runs were conducted for each test case and the worst, best, average performance, the standard deviation of the run results, the improvement of the average performance against the actual operation and the average running times are shown. The actual and MIP results in Tables 1 and 2 are also listed here for easy reference. We also validate the GA's performance via the open source code provided by Martello et al., which is downloadable at http://www.diku.dk/˜pisinger/new3dbpp/3dbpp.c. The Martello et al. code can solve the general 3D-BPP with fixed carton orientations. We validate the GA solutions by selecting one of the best solutions found by the GA and feeding the orientation information to the code and trying to pack all the cartons into a bin with the corresponding height. The code is allowed to run for 2 hours. If the cartons can be packed by the code, then a 'Yes' is marked under the column 'Packed', otherwise a 'No' is filled. The time used (in seconds) by the code until it exits is also listed in the tables.

Table 3
GA performance for small bin test cases

			GA						Martello et al.[*]	
Case	Act.	MIP	Worst	Best	Avg	Std	Iprv(%)	T (s)	Packed	T (s)
SM01	91	80	82	80	81.2	1.01	10.77	0.56	Yes	0.1
SM02	56	37	38	37	37.3	0.44	33.39	0.74	Yes	0.1
SM03	69	47	47	47	47.0	0.00	31.88	0.79	Yes	0.1
SM04	91	82	91	83	85.2	2.74	6.37	1.59	Yes	0.6
SM05	56	52	58	52	55.5	1.93	0.89	0.82	Yes	1.8
SM06	53	38	47	40	44.5	3.20	16.04	1.48	Yes	0.1
SM07	70	62	62	62	62.0	0.00	11.43	2.82	Yes	3.0
SM08	91	71	71	69	69.8	0.89	23.30	2.47	Yes	0.1
SM09	71	62	62	52	58.3	3.88	17.89	3.96	No	7200.0
SM10	85	62	70	63	66.1	1.96	22.24	3.89	No	7200.0

[*]General 3D-BPP code: http://www.diku.dk/˜pisinger/new3dbpp/3dbpp.c

For small bin test cases, the GA worst performance outperforms the actual packing results except for SM05; however, the GA worst performance is inferior to MIP results except for SM03, SM07, SM08 and SM09. The GA best performance achieves equivalent or better solutions compare to the MIP results except for SM04, SM06 and SM10. The GA average performance is close to or better than the MIP performance for SM07 to SM09, but is outperformed by MIP for SM01, SM02, SM04, SM05, SM06 and SM10. This indicates that the MIP model works well for small test cases. The code of Martello et al. can pack the small bins (SM01 to SM08) with very short time if given the cartons orientation, but fails to validate the solutions for SM09 and SM10 within 2 hours' time.

For big bin test cases shown in Table 4, it is observed that GA worst performance outperforms the actual results in all cases; performance for some cases is significantly improved. Compare with the MIP results, GA worst is only outperformed by MIP in 5 cases. The GA best performance provide equivalent or better solutions than the MIP does in all cases except BM02 and BM04. The GA average also provides satisfactory performance compared with the MIP results. It should be pointed out that as the number of cartons to be packed increases, the GA tends to outperform the MIP. The validation through the code of Martello et al. indicate that for most cases, the code is not able to find a solution to pack the cartons even the cartons orientations are given. This indicates the difficulty of the test cases used in this paper; on the other hand, this is may due to the big size difference of cartons used for the test cases which might not exploit the features of the Martello et al. code.

Table 4
GA performance for big bin test cases

			GA						Martello et al.[*]	
Case	Act.	MIP	Worst	Best	Avg	Std	Iprv(%)	T (s)	Packed	T (s)
BM01	91	84.1	84.1	84.1	84.1	0.00	7.58	1.2	Yes	0.1
BM02	91	83.1	86.4	84.1	84.2	0.51	7.47	4.8	No	7200.0
BM03	45	32	32	32	32.0	0.00	28.89	7.1	Yes	0.1
BM04	91	77.4	84.1	83.4	83.5	0.26	8.24	5.4	Yes	5.0
BM05	91	79	83.1	79	81.8	1.59	10.11	6.5	No	7200.0
BM06	67	40	47	40	44.3	2.49	33.88	12.7	Yes	0.1
BM07	48	40	40	40	40.0	0.00	16.67	23.6	Yes	6.0
BM08	59	53	53	51	52.6	0.82	10.85	24.7	No	7200.0
BM09	94	82	74	72	73.7	0.73	21.60	20.4	No	7200.0
BM10	91	63.1	61	61	61.0	0.00	32.97	23.8	Yes	0.6
BM11	91	78	77.4	77.4	77.4	0.00	14.95	24.4	No	7200.0
BM12	94	68	61	61	61.0	0.00	35.11	32.4	Yes	158.8
BM13	83	78	77	71	74.9	2.47	9.76	28.4	No	7200.0
BM14	95	88.4	85	78.4	81.7	2.04	14.11	27.2	Yes	205.1
BM15	91	87	78	75	77.6	0.76	14.73	33.0	No	7200.0
BM16	63	52	52	48	50.5	1.96	19.84	49.6	No	7200.0
BM17	91	88.4	85	80	82.9	1.60	8.90	30.5	No	7200.0
BM18	70	73.4	68	63.1	65.8	1.89	6.00	40.8	No	7200.0
BM19	94	79	68	63.1	66.2	1.62	29.57	40.0	No	7200.0
BM20	94	87	84	79	80.8	1.24	14.04	32.9	No	7200.0
BM21	95	87	84.1	78.4	80.6	1.95	15.16	33.1	No	7200.0
BM22	91	87	78	78	78.0	0.00	14.29	34.2	No	7200.0
BM23	95	93.4	82	78.4	79.4	1.01	16.42	38.5	No	7200.0
BM24	91	89	90	80	86.0	3.07	5.49	44.8	No	7200.0
BM25	94	79	78	76	77.6	0.60	17.45	53.6	No	7200.0
BM26	91	97	87	82	85.9	1.09	5.60	50.8	No	7200.0
BM27	91	109	90	85	87.2	1.36	4.18	59.3	No	7200.0

[*]General 3D-BPP code: http://www.diku.dk/˜pisinger/new3dbpp/3dbpp.c

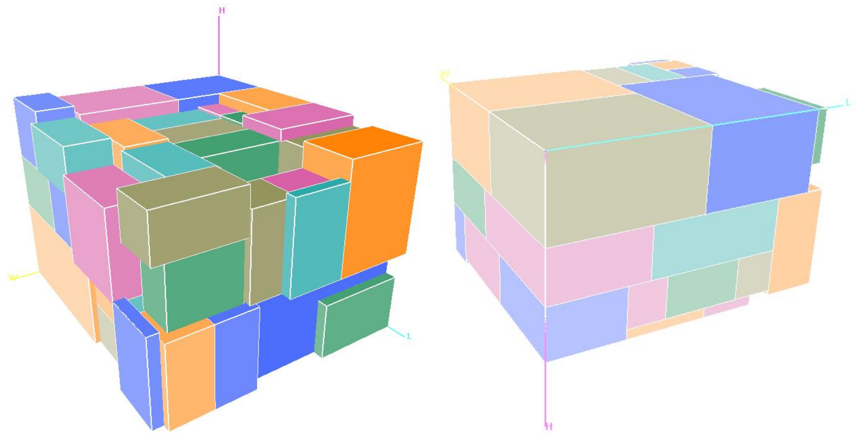
(a) Top view (b) Bottom view

Fig. 2. Packing results achieved for a big bin.

Figure 2 presents the packing results for a big bin with views from the top and the bottom of the bin, where it can be observed that the cartons are compactly packed.

5 Batch bin packing

The batch bin packing problem is the extension of the single bin packing problem we have discussed so far. The problem is to ship a batch of cartons with the variable height bins available so that the ultimate shipping cost/volume can be minimized.

For single bin packing problem, the cartons and the bin are fixed. Therefore, there is no need to consider the space charged for the pallet leg and cushion layer; the bin height is the sole goal to pursue. However, in a batch bin packing problem, the types of bins to be used must consider the pallet legs and cushion layers as different type of bin has different volume for pallet leg and cushion layer. Therefore, the objective function needs to be changed to:

$$\min \sum_i L_1 W_1(\tilde{H}_i + H_{L_1} + H_{C_1}) + \sum_j L_2 W_2(\tilde{H}_j + H_{L_2} + H_{C_2}) \tag{6}$$

The objective function calculates the overall volume to be charged by the logistics service providers, where L_1, W_1 and L_2, W_2 are the length and width of small and big bins, respectively; H_{L_1}, H_{C_1}, H_{L_2} and H_{C_2} are the height for the pallet leg and cushion layer for small and big bins. For the batch bin packing problem, we use the heuristic shown in Algorithm 2.

Algorithm 2 presents a heuristic approach based on the solution proposed in Section 4 for single bin packing. It utilizes a two-stage approach: the first stage tries to

Algorithm 2 Batch bin packing algorithm

```
1: Read in all the carton information and create a list $L_m$ to store all the cartons;
2: while the overall carton volume of $L_m$ exceeds the threshold $\alpha$ do
3:    Randomly pick cartons from $L_m$ and put into list $L_t$ until the overall carton volume of $L_t$ exceeds $\alpha$;
4:    Use the single bin packing algorithm to pack $L_t$ and get the height of the bin;
5:    if the height does not exceed the height limit then
6:       Randomly select cartons from $L_m$ and put into list $L_a$ so that the total carton volume of $L_a$ is about the gap capacity remaining for the bin;
7:       Combine $L_a$ with $L_t$;
8:       while the combined list cannot be put in the bin do
9:          Randomly remove one carton out of $L_t$ and put it back into $L_m$;
10:         Try to pack the cartons into the bin;
11:      end while
12:   else
13:      repeat
14:         Randomly remove one carton out of $L_t$ and put it back into $L_m$;
15:         Try to pack $L_t$ into the bin;
16:      until $L_t$ can be packed in the bin
17:   end if
18:   Set $L_t = \varnothing$, $L_a = \varnothing$;
19: end while
20: Evaluate the remaining cartons with three cases: one big bin, one small bin and two small bins;
21: Select the feasible case where the overall bin volume is minimized for the remaining cartons;
22: Output the results.
```

pack cartons into big bins if the overall remaining carton volume exceeds a threshold α, which is a conservative fill rate estimation, e.g., 75% of the capacity of a big bin. The second stage evaluates the three scenarios where a big bin, a small bin and two small bins are used and the scenario with the minimal overall volume calculated according to (6) for the stage two cartons is selected. The underlying principle is to use as many big bins as possible while minimizing the overall volume used for the stage two cartons.

We use the 37 test cases to create 3 batches to test the performance of the proposed batch bin packing algorithm. The first batch randomly picks 5 big bin test cases and 5 small bin test cases; the second batch includes 10 randomly selected big bin test cases and the third batch uses the remaining 17 test cases.

Table 5 reports the net fill rate and gross fill rate for the actual results, the results from individual GA runs and the batch bin packing runs. The net fill rate for 'Actual' is calculated using the actual height of each bin; the individual GA uses the

Table 5
Performance for batch bin packing.

		Net fill rate			Gross fill rate		
Batch	Cartons	Actual	Ind. GA	Batch GA	Actual	Ind. GA	Batch GA
1	217	72.91%	82.70%	85.39%	57.07%	63.42%	68.05%
2	262	68.06%	85.07%	85.47%	54.27%	65.32%	68.07%
3	338	71.01%	82.83%	84.38%	56.58%	64.40%	67.35%

average performance shown in Tables 3 and 4 as the bin height and the batch GA is the average of 20 runs for each batch; only the usable bin volume is used to calculate the net fill rate. For gross fill rate, the volume for the pallet leg, cushion layer and shrink wrapping is taken into consideration.

It can be observed from Table 5 that the net fill rates increase for all 3 batches when applying the individual GA; the fill rates are further improved when using the batch bin packing algorithm. The maximum fill rate improvement is 17.41% for batch 2 and minimum improvement is 12.48% for batch 1. For the gross fill rate, the similar pattern can be obtained with a minimum improvement of 10.98% and a maximum improvement of 13.8%. The difference between the individual GA runs and batch GA runs indicates pooling cartons together and packing in batches will be helpful for the actual operations.

6 Conclusion

This paper discussed the 3D-BPP with variable bin heights. The cartons to be packed can be freely rotated into different orientations. A mathematical model was modified based on the literature and numerical experiments were conducted. The results indicated that the mathematical approach is still not suitable for real time bin packing problems. A special bin packing algorithm based on packing index was designed to utilize the special problem feature and was used as a building block for a genetic algorithm designed for the variable height 3D-BPP. Subsequently, the GA was applied to the situation where more than one type of bin can be used and a heuristic for packing a batch of cartons was designed. The performance of the GA was compared with the actual operation and the mathematical model; an open source general bin packing algorithm was also used to validate the results. Numerical experiments indicated that the proposed GA can improve the packing results significantly compared to the actual operations and provides quick and satisfactory results.

Acknowledgement

The authors would like to sincerely thank the three anonymous reviewers for their pertinent comments and suggestions for this paper, which not only improved the quality, but also the presentation of this paper. The authors want to thank Emerson Singapore Distribution Center for providing the operational benchmark data, and special thanks to Mr. Jack Wong and Mr. Andy Chang.

References

Alves, C. and de Carvalho, J. M. V. (2007). Accelerating column generation for variable sized bin-packing problems. *European Journal of Operational Research*, 183:1333–1352.

Baltacioğlu, E., Moore, J. T., and Hill Jr., R. R. (2006). The distributor's three-dimensional pallet-packing problem: a human intelligence-based heuristic approach. *International Journal of Operational Research*, 1(3):249–266.

Bischoff, E. E., Janetz, F., and Ratcliff, M. S. W. (1995). Loading pallets with non-identical items. *European Journal of Operational Research*, 84:681–692.

Bischoff, E. E. and Marriott, M. D. (1990). A comparative evaluation of heuristics for container loading. *European Journal of Operational Research*, 44:267–276.

Chen, C. S., Lee, S. M., and Shen, Q. S. (1995). An analytical model for the contianer loading problem. *European Journal of Operational Research*, 80(1):68–76.

Correia, I., Gouveia, L., and da Gama, F. S. (2008). Solving the variable size bin packing problem with discretized formulations. *Computers & Operations Research*, 35:2103–2113.

Crainic, T. G., Perboli, G., and Tadei, R. (2008). Extreme point-based heuristics for three-dimensional bin packing. *INFORMS Journal on Computing*, 20(3):368–384.

den Boef, E., Korst, J., Martello, S., Pisinger, D., and Vigo, D. (2005). Erratum to "The three-dimensional bin packing problem": Robot-packable and orthogonal variants of packing problems. *Operations Research*, 53(4):735–736.

Dósa, G. (2007). The tight bound of first fit decreasing bin-packing algorithm is $FFD(I) \leqslant 11/9\, OPT(I) + 6/9$. In *Combinatorics, Algorithms, Probabilistic and Experimental Methodologies*, volume 4614 of LNCS, pages 1–11. Springer, Berlin.

Egeblad, J. and Pisinger, D. (2009). Heuristic approaches for the two- and three-dimensional knapsack packing problem. *Computers & Operations Research*, 36:1026–1049.

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

Gehring, H. and Bortfeldt, A. (1997). A genetic algorithm for solving the container loading problem. *International Transactions in Operational Research*, 4:401–418.

George, J. A. (1996). Multiple container packing: A case study of pipe packing. *The Journal of the Operational Research Society*, 47(9):1098–1109.

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

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Kluwer Academic Publishers, Boston, MA.

Hopper, E. and Turton, B. C. H. (2001). An empirical investigation of meta-heuristics and heuristic algorithms for a 2D packing problem. *European Journal of Operations Research*, 128:34–57.

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

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

Martello, S., Pisinger, D., Vigo, D., den Boef, E., and Korst, J. (2007). Algorithm 864: General and robot-packable variants of the three-dimensional bin packing problem. *ACM Transactions on Mathematical Software*, 33(1):1–12.

Michalewicz, Z. (1994). *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

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

Puchinger, J. and Raidl, G. R. (2007). Models and algorithms for three-stage two-dimensional bin packing. *European Journal of Operational Research*, 183:1304–1327.

Puchinger, J., Raidl, G. R., and Koller, G. (2004). Solving a real-world glass cutting problem. In *Evolutionary Computation in Combinatorial Optimization – EvoCOP 2004*, volume 3004 of LNCS. Springer.

Zhang, D.-F., Wei, L.-J., Chen, Q.-S., and Chen, H.-W. (2007). A combinatorial heuristic algorithm for the three-dimensional packing problem (in Chinese with English abstract). *Journal of Software*, 18(9):2083–2089.