2006 IEEE Congress on Evolutionary Computation
Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada
July 16-21, 2006

On Solving Multiobjective Bin Packing Problems Using Particle Swarm Optimization

D. S. Liu, K. C. Tan, C. K. Goh and W. K. Ho

Abstract— **The bin packing problem is widely found in applications such as loading of tractor trailer trucks, cargo airplanes and ships, where a balanced load provides better fuel efficiency and safer ride. In these applications, there are often conflicting criteria to be satisfied, i.e., to minimize the bins used and to balance the load of each bin, subject to a number of practical constraints. Unlike existing studies that consider only the minimization of bins, a two-objective mathematical model for the bin packing problem with multiple constraints is formulated in this paper. Without the need of combining both objectives into a composite scalar, a hybrid multiobjective particle swarm optimization algorithm (HMOPSO) incorporating the concept of Pareto's optimality to evolve a family of solutions along the trade-off is proposed. The algorithm is also featured with bin packing heuristic, variable length representation, and specialized mutation operator to solve the multiobjective and multi-model combinatorial bin packing problem. Extensive simulations are performed on various test instances, and their performances are compared both quantitatively and statistically with other optimization methods. Each of the proposed features is also explicitly examined to illustrate their usefulness in solving the multiobjective bin packing problem.**

I. Introduction

The bin packing problem is a NP-hard combinatorial optimization problem where the primary aim is to pack a finite number of items using the least bins possible. It involves a number of practical constraints and requirements to be satisfied, such as weight, centre of gravity, irregularly shaped bins and priority items. The bin packing problem has been widely studied due to its various applications in computer network design [5], job scheduling on uniform processors [28], industrial balancing problem [3], aircraft container loading [20] etc.

Many computer-assisted approaches have been developed to solve the bin packing problem. Regarding waste space minimization for one to three dimensional bin packing problems, exact solution methods based on branch and bound procedure are proposed in [18], [19], [24]. However, these methods are more applicable for moderate and/or small problems.

Besides the branch and bound method, heuristic approaches are also widely used for solving the bin packing problem. The heuristic methods are particularly useful for problems with a high complexity, for which deterministic methods like branch and bound approach are often unable to find the solution within a reasonable amount of time. For example, methods such as next-fit, first-fit, and best-fit heuristics have been successfully applied to bin packing problems [1], [2].

Metaheuristics such as evolutionary algorithms [8], [11]-[16], [21]-[23], [25], [26], simulated annealing [4] and tabu search [17] have also been widely applied to solve the combinatorial bin packing optimization problem. Among these methods, the evolutionary algorithm [6] is usually hybridized with a heuristic placement approach for solving the bin packing problem. The sequence of the item to be packed is usually constructed as a sequence chromosome, which is decoded via a packing heuristic for generating a packing plan to be evaluated against the objective function to obtain the quality of the solution.

In this paper, a multi-objective (MO) two-dimensional bin packing model is formulated, which provides a good representation for the multiobjective real-world bin packing problem, such as the problem of loading tractor trailer trucks. In this problem, the trucks are subject to many constraints and one of these constraints regulates the maximum weight per axle that can be carried by one of these trucks. The solution to MO problem exists in the form of alternate tradeoffs known as Pareto optimal set. It is thus important to develop an effective and efficient algorithm capable of finding the set of optimal packing solutions that utilize the maximum possible volume of each trailer and minimize the per axle weight.

Particle swarm optimization (PSO), a relatively new computational intelligence technique, is based on swarming theory where all particles in the neighborhood depend on their discoveries and past experiences of their neighbors. Social sharing of information offers an evolutionary advantage and aids towards solving a difficult problem like bin packing problem. Although PSO is relatively new, it has been shown to offer higher convergence speed for MO optimization as compared to canonical multiobjective evolutionary algorithms (MOEAs). PSO has never been considered in the context of solving bin packing problem. In this paper, a hybrid multiobjective particle swarm optimization algorithm (HMOPSO) is proposed to solve the two objective bin packing problem mentioned above. With the real multiobjective algorithm HMOPSO, the packing solution generated can be used for resource planning and decision making and satisfy different customer's different requirement.

The paper is organized as follows: Section II describes the

D. S. Liu, K. C. Tan, C. K. Goh and W. K. Ho are with the Department of Electrical and Computer Engineering, National University of Singapore, 4, Engineering Drive 3, Singapore 117576. Email: eletankc@nus.edu.sg

0-7803-9487-9/06/$20.00/©2006 IEEE

scenario and modeling of the multiobjective bin packing problem. Section III describes the proposed HMOPSO algorithm and its various features including variable-length representation, specialized mutation operators, and Pareto fitness ranking are also described. Section IV gives an analysis on the performance of HMOPSO algorithm in terms of the simulation results. A comparison on the performance of HMOPSO and another two algorithms are also provided to show the effectiveness and efficiency of HMOPSO in solving multiobjective bin packing problems.

II. Problem Formulation

The proposed two objective two-dimensional bin packing problem (MOBPP-2D) is defined as follows: Given a maximum of I bins with width W, height H, and J items with $w_j \le W$, $h_j \le H$ and weight ψ_j, pack the items into bins such that the number of bins used K is minimized and the overall centre of gravity of the bins is as close as possible to the desired CG. The second objective – the minimization of the average deviation of CG of all the bins towards the idealized CG (CG_{dev}), ensures that the bins are stable and the overall CG of the bins is low.

The proposed mathematical model for MOBPP-2D is as follows:

$$\text{Number of bins used } K = \sum_{i=1}^{I} \left\lceil \sum_{j=1}^{J} \frac{X_{ij} h_j w_j}{HW} \right\rceil \tag{1}$$

Average deviation of CG from idealized CG of bins

$$CG_{dev} = \frac{1}{K} \sum_{i=1}^{K} \sqrt{(\Phi_{x,i} - \Phi_{x,d})^2 + (\Phi_{y,i} - \Phi_{y,d})^2} \tag{2}$$

where

$$X_{ij} = \begin{cases} 1 & \text{if item } j \text{ is assigned to bin } i \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

$$\Phi_{x,i} = \sum_{j=1}^{J} X_{ij} x_j \psi_j \Big/ \sum_{j=1}^{J} \psi_j \tag{4}$$

$$\Phi_{y,i} = \sum_{j=1}^{J} X_{ij} y_j \psi_j \Big/ \sum_{j=1}^{J} \psi_j \tag{5}$$

$\Phi_{x,i}$ is center of gravity of bin i in x direction; $\Phi_{y,i}$ is center of gravity of bin i in y direction; $\Phi_{x,d}$ is desired center of gravity of bin i in x direction; $\Phi_{y,d}$ is desired center of gravity of bin i in y direction; Each item must be assigned to one bin and only to one bin $\sum_{i=1}^{K} \sum_{j=1}^{J} X_{ij} = J$.

A. Importance of the Stability Objective

In this paper, we considered the CG constraint as a means to obtain a minimum number of stabilized bins and the stabilizing of the bin is done by minimizing the deviation of the overall CG of the loaded bin from the desired CG of the bin. The desired CG is assumed as $(W/2, 0)$. The bin is considered stable if the deviation is small.

If the deviation of CG is small, the bins could be prevented from toppling when a force is applied to the side of the bin. In container loading problem, the toppling of the container could result in time wastage and high cost in storage and transportation. Balanced load also plays an important role in application such as loading of tractor trailer trucks and cargo airplanes. In loading trailers and trucks, a balanced load can minimize the maximum weight per axle and provide better fuel efficiency and better response of the truck as a whole to the loading. Besides, trucks have to maintain their centers of gravity low enough to withstand any tendency to topple over when driven around sharp corners. In aircraft loading, the arrangement of the cargo affects the position of the center of gravity of the aircraft, which in turn has an impact on aircraft drag. One therefore wants to balance the load so that the aircraft will fly more safely, fly faster and use less fuel.

B. Test Case Generation

The test cases considered in the MOBPP-2D is randomly generated to investigate how the CG objective is going to create stabilized bins while packing all the items efficiently into the bins. 6 classes with 20 instances each are being randomly generated with four set of items with different sizes. It is assumed that the item is small enough to be packed into the bins ($W = H = 100$ for each bin). It is also assumed that there is no weight limit for the bins used.

Class 1: w_j and h_j randomly generated in the range [0, 100], ψ_j in the range of [0, 20] for the first 10 instances and ψ_j in the range of [0, 100] for the next 10 instances.

Class 2: w_j and h_j randomly generated in the range [0, 25], ψ_j in the range of [0, 20] for the first 10 instances and ψ_j in the range of [0, 100] for the next 10 instances.

Class 3: w_j and h_j randomly generated in the range [0, 50], ψ_j in the range of [0, 20] for the first 10 instances and ψ_j in the range of [0, 100] for the next 10 instances.

Class 4: w_j and h_j randomly generated in the range [0, 75], ψ_j in the range of [0, 20] for the first 10 instances and ψ_j in the range of [0, 100] for the next 10 instances.

Class 5: w_j and h_j randomly generated in the range [25, 75], ψ_j in the range of [0, 20] for the first 10 instances and ψ_j in the range of [0, 100] for the next 10 instances.

Class 6: w_j and h_j randomly generated in the range [25, 50], ψ_j in the range of [0, 20] for the first 10 instances and ψ_j in the range of [0, 100] for the next 10 instances.

For all the instances generated, there are 500 items to be packed. From the data set generated, Class 2 seems to be more difficult than the rest of the classes since there are a large number of small items. Since all the bins used in all the classes are of the same sizes, a larger number of permutation is needed to achieve a more stable configuration, whereas for larger items, there is lesser permutation to be arranged to achieve a stable configuration.

III. Hybrid Particle Swarm Optimization

In this section, a hybrid multiobjective particle swarm

optimization (HMOPSO) algorithm for solving multi-objective bin packing problem is presented.

A. *Bottom Left Fill Heuristic*

An important feature of HMOPSO is the choice of a good heuristic to build the solutions. Generally heuristic approaches for solving the two-dimensional bin packing problem are either level-oriented or bottom-left oriented. Berkey et al. [2] have successfully applied heuristic such as level-oriented next-fit, level-oriented first-fit and next bottom-left on solving two-dimensional bin packing problem. The bottom-left (BL) procedure has also been employed by [13], [16] in genetic algorithms and demonstrates good results. However, BL-routine tends to generate layouts with relatively large empty areas. Therefore, HMOPSO employs the bottom-left-fill algorithm (BLF) [11] that is capable of filling existing gaps in the partial packing layout.

The BLF heuristic adheres strictly to the BL condition. The packing configuration which is generated fulfills the BL condition if no rectangle packed can be moved further to the bottom and to the left [13]. The BLF heuristic packs rectangles by searching for available space in the bottom left manner. A list of insertion points is maintained to allow the algorithm to search for space to insert a rectangle. New insertion points are inserted into the list whenever a rectangle is placed at an insertion point and the insertion point at which the rectangle is placed is deleted from the list. An intersection test is carried out for rectangle at every insertion point in the list with those already in the bins.

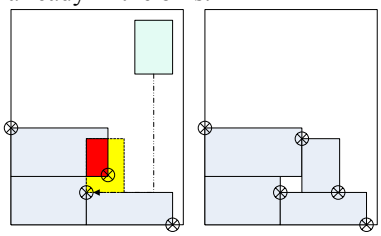


Figure 1: The insertion at new position when an intersection is detected at the top

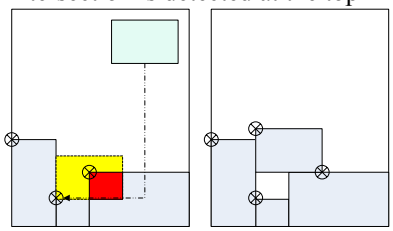


Figure 2: The insertion at new position when an intersection is detected at the right

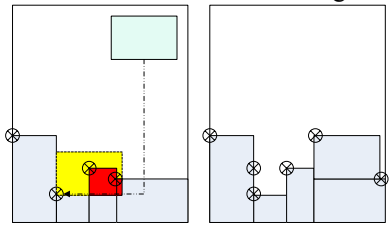


Figure 3: The insertion at next lower position with generation of three new insertion points

The item is moved to the right if there is an intersection detected at the top (shown in Figure 1) and to the top if there is an intersection detected at the right (shown in Figure 2). The empty space is still available for the insertion of items if the items can fit in. As shown from Figure 3, if there is a lower available space for insertion, the rectangle is inserted there instead at the newly generated position. A new bin is generated if there is no space in previous bins to accommodate a new rectangle. The BLF heuristic in pseudo code is presented below.

```
For each rectangle in list
  For each available bin
    For each insertion point in the insertion list
      check for intersection
      If there is an intersection
        insert a new insertion point
      else
        insert rectangle at current insertion point
        remove the current insertion point
        insert two new insertion point at the top left and bottom right of current rectangle
        sort the insertion list in bottom left order
      End If
    End For
  End For
  If there is no bins which can accommodate the rectangle
    create a new bin
    insert the rectangle in the newly created bin
  End If
End For
```

B. *Variable-Length Representation of the Particle*

Each particle is represented by a variable length representation which encodes the number of bins used and the order of which the rectangles are packed into the bin by the BLF heuristic. In every bin, there must be at least one rectangle and that rectangle must not be found in any other bins meaning that the permutation encoded are unique for every bin. As shown in Figure 4, a particle consists of several bins and each bin holds a number of rectangles where the number is not constant and is restricted by the capacity of the bin. Each bin also encodes the sequence of the rectangles to be packed into the bin by the BLF algorithm.

In HMOPSO, the *pbest* is defined as the particles' own best bin due to their own best values in the solution space. The *gbest* is defined as the best bin of another non-dominated particle in the archive. The *pbest* and *gbest* are also encoded in the form of extra bins which is stored in the particle memory.

Such a variable length representation is versatile as it allows the algorithm to manipulate the permutation of rectangles in each bin directly for optimization without affecting other bins. And BLF check can be applied only to the changed bins. However, the permutation encodes the item only by its index and not the properties of the item. A new state variable is attached to every index in the permutation to allow the variable length representation to be able to encode both the index and the properties of the item. In this bin packing problem, since the extra property to be encoded is the orientation of the rectangle, a negative sign is proposed to be added to the permutation to allow easy manipulation. In Figure 4, the negative sign of the index of the rectangle encodes the orientation of the rectangle where the presence of

the negative sign indicates that the rectangle is rotated by 90 degrees and the absence indicates that the rectangle is not rotated. With this new feature, the mutation operators have to be redesigned to manipulate the sequence to allow important property of the item encoded in the permutation to be inherited.

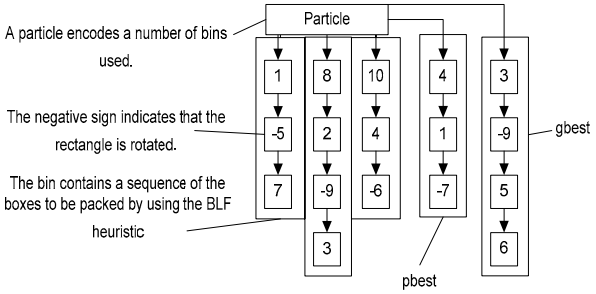


Figure 4: The data structure of particle representation

C. Initialization

As the search space of the bin packing problem is very large, the building of the initial population becomes important as it is responsible for initializing the particle with starting solutions which are scattered throughout the search space. To allow the swarm of particles to be initialized with good solution, some of the sequences are randomly generated whereas some are initially sorted according to some criteria, e.g. width, height, area and perimeter based on the list of rectangles to be packed. To allow more variation in the sorted sequences, some are further partial-randomized to avoid having repeated sequences. With the generated sequences, the list of bins is built using the BLF heuristic. The whole procedure of initialization is described in Figure 5.

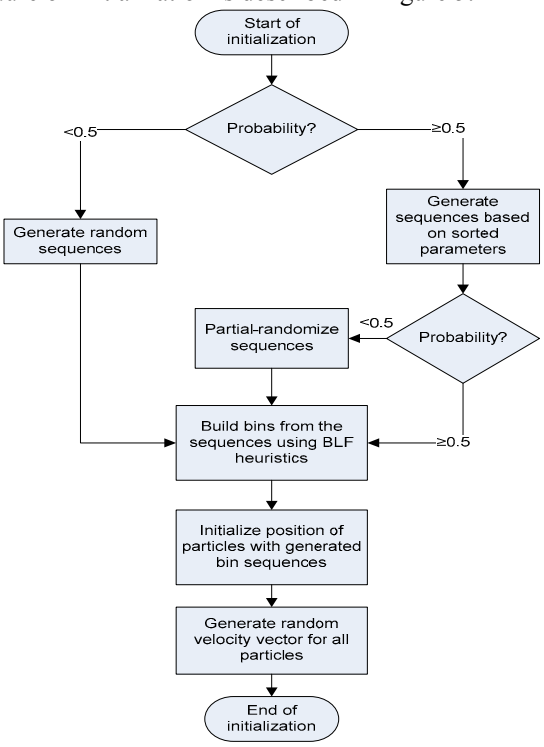


Figure 5: Initialization for the swarm of particles

D. Implementation of the PSO operator

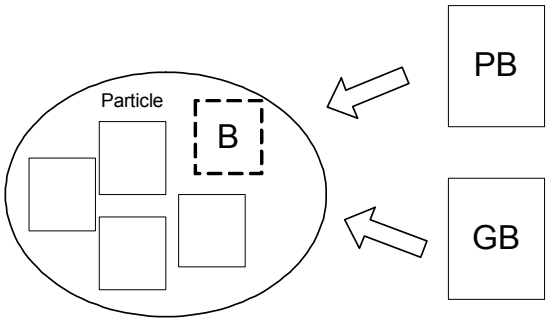
In this paper, the velocity is either governed by *pbest* or *gbest* but not both at any one time as shown in equation 6. The velocity is then used in the determination of the next position to move to in the solution space where this movement is represented by equation 6.

$$V_{id} = \alpha * P_{id} \oplus (1-\alpha) * P_{gd},\ \alpha = 0,\ 1 \quad (6)$$

$$X_{id} = X_{id} + V_{id} \quad (7)$$

As shown in Figure 6, each particle is updated inserting the bin represented by the velocity vector into the particle and deleting any duplicate item in other bins.

The *pbest* of the particle is updated by evaluating the current solution with its previous best solution to see whether the current solution dominates the previous best solution. The *pbest* is replaced with the current best bin and the best solution is replaced by the current solution if the current solution dominates the previous best solution. If both solutions are non-dominating to each other, there is a 50% probability of the *pbest* being replaced by the current best bin and the previous best solution replaced by the current solution. The selection of *gbest* is by the tournament niche method where a particle with a lower niche count is selected out of two randomly selected non-dominated particles from the archive.



B: Can be either PB or GB but not both at the same time
PB: The most filled bin for pbest
GB: The most filled bin for gbest

Figure 6: PSO Mechanism for updating position of particle

E. Specialized Mutation Operator

With the new variable length representation, specialized mutation operators have to be constructed to allow the operators to manipulate the new data representation. Since the new representation also encodes the orientation property of the item, the rotation mutation operator is designed specially to mutate the orientation of the rectangle.

The specialized mutation mechanism is a three-stage mutation process which provides the facilities to manipulate the internal bins represented by the particle. As shown in Figure 7, in the first stage, the particle will undergo either 1) partial swap where the sequence in two bins are randomly cut and exchanged between the two bins, 2) merge bins process which chooses two least filled bins and merge the items in them into one bin, or 3) the split bin process which randomly split the contents in the bin into two bins. In the second and

third stage, the particle undergoes 1) random rotation of the rectangles in the bin and 2) random shuffle of bin contents to try to search for an arrangement which can improve the packing configuration and the stability of the bins respectively. A feasibility check is then applied to make sure all the constraints are satisfied. If any constraint is violated, no mutation takes place and the particle reverts back to its original position. This avoids the need to implement a repair mechanism to repair the particle if any constraint is violated.

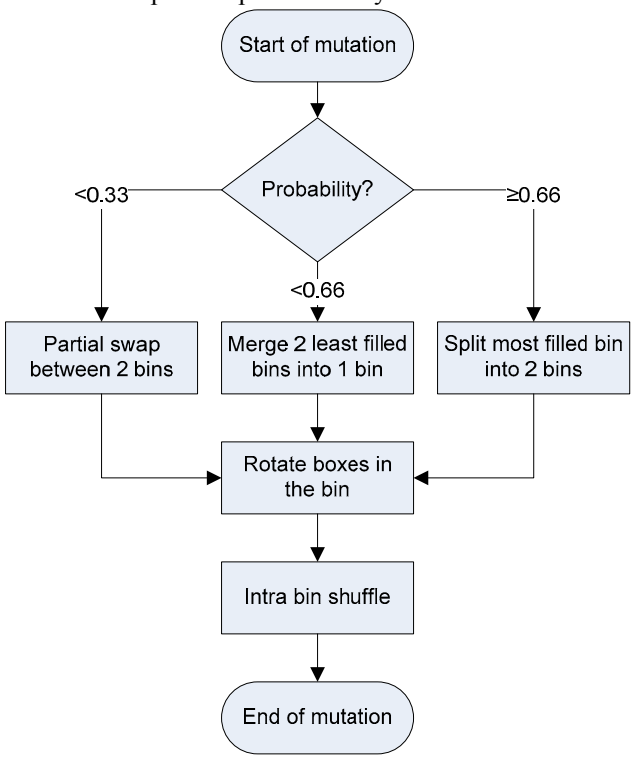


Figure 7: Mutation modes for a single particle

F. *Flowchart of HMOPSO*

The flowchart of HMOPSO is illustrated in Figure 8. Before HMOPSO can manipulate the data representation of the bin packing problem, the building of the initial solutions into a format that can be used becomes the first step of the algorithm. Once the initialization has finished, the particles are evaluated against the fitness function and ranked accordingly by the Pareto ranking scheme [9]. Simple fitness sharing [10] is applied to distribute the solutions along the Pareto front. Tournament selection [27] with a size of 2 is then applied to select the particles for movement to a new position in the solution space. The particles are randomly grouped into pair of two and those particles with a lower rank in partial order are then selected. All the particles that remained are then updated with new velocity vectors by selecting either their *gbest* or *pbest* as the velocity vector. Their positions in the solution space are then updated with the velocity vectors by inserting the bin represented by the velocity vector into the particle. At their new positions, the particles then undergo the specialized mutation operators. After that, all the non-dominated solutions are inserted into the archive for storage and for the evaluation of *gbest* in future generations. The evolution progresses through a predefined number of generations and stops only when the predefined number of generation is reached.

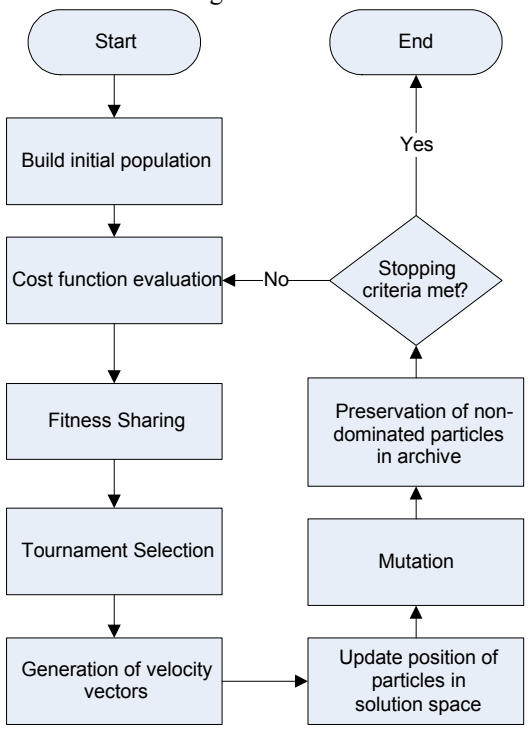


Figure 8: The flowchart of HMOPSO

IV. Simulation Results

The HMOPSO was programmed in C++ based on a Pentium IV 2.8 GHz processor with 512 MB RAM under Microsoft Windows XP. In order to evaluate the effectiveness of HMOPSO, two algorithms [27], [30] are implemented to solve the bin packing problem. The PSO in [30] has been successfully applied to the traveling salesman problem while the MOEA presented in [27] has been designed for solving truck and trailer vehicle routing problem. Both algorithms have been adapted for the bin-packing problem. The parameters settings used in this part of the analysis are shown in Table 1.

Table 1: Parameter settings used by MOPSO, MOEA and HMOPSO

Parameter	MOPSO	MOEA	HMOPSO
crossover rate	-	0.7	-
PSO update rate	-	-	0.7
mutation rate	-	0.4	0.4
population size	500	500	500
generation size	200 generations		
niche radius	0.1	0.1	0.1

A. *Characteristics of Different Problems*

The test cases are designed to examine situations when different combinations of items with different sizes are grouped together. Items which are less than a quarter of the size of the bin are considered small. Items which are larger

than a quarter but less than half size of the bin are considered to be medium size items. Items which are larger than half size of the bin are considered large. The stability of the bin is one of the objectives in MOBPP-2D. This section examines how the combinations of different sized items are going to affect the stability of the bins when these items are packed into the bin.

Class 2 consists of only small items. Class 3 consists of a combination of small and medium sized items whereas class 4 consists of a combination of small, medium and large items. Figure 9 shows the box plot of average deviation of CG for the non-dominated solutions in 9 randomly selected test cases. From the figure, it can be observed that the mean deviation of CG of class 2 is consistently lower than class 3 and class 4 with the test cases class 4 scoring the highest mean deviation of CG. The result shows that when the items are small, there are more rooms to arrange the items to achieve lower stability in the bins. When there is a mixture of small and medium size items, there appears to be less permutation to arrange. The worst case is when there is a mixture of small, medium and large items to arrange. However, with less permutation to arrange, the problem becomes much easier where convergence is achieved easily. Therefore, the class 2 test cases appear to be the hardest among the 6 classes. From the result, it could be seen that the average deviation of CG of most solutions generated are less than half the height of the bin. This further shows that HMOPSO is capable in generating solutions which uses a lesser number of stabilized bins.

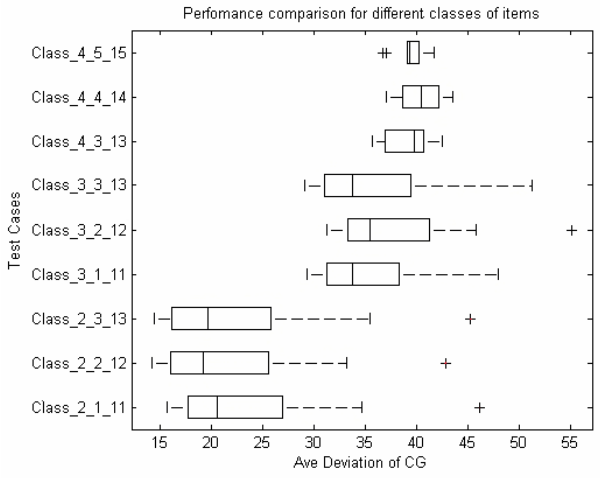


Figure 9: Average deviation for different classes of items

B. SO Optimization Performance

A comparison on the performance of metaheuristic algorithms against the branch and bound method (BB) on single objective bin packing problem is done by running the algorithms against the 6 classes of test cases from [18]. Table 2 shows the average number of optimal solutions obtained by each algorithm in 10 runs. From the results, it can be observed that the proposed method is capable of evolving more optimal solution as compared to BB in 5 out of 6 classes of test instances.

Table 2: Number of optimal solution obtained by branch and bound method, MOEA, MOPSO and HMOPSO

Class 1	**BB**	**MOEA**	**MOPSO**	**HMOPSO**
n=20	10	10	10	10
n=40	10	10	9	10
n=60	7	6	5	6
n=80	3	10	5	10
n=100	1	6	3	7
Total	31	**42**	**32**	**43**
Class 2				
n=20	10	10	10	10
n=40	10	9	9	9
n=60	4	10	9	10
n=80	10	9	8	9
n=100	10	10	9	10
Total	44	**48**	**45**	**48**
Class 3				
n=20	9	10	10	10
n=40	9	6	5	6
n=60	5	5	1	6
n=80	0	4	1	4
n=100	0	5	1	5
Total	23	**30**	**18**	**31**
Class 4				
n=20	10	10	10	10
n=40	10	10	10	10
n=60	7	8	8	8
n=80	10	7	7	7
n=100	10	8	7	8
Total	47	43	42	43
Class 5				
n=20	10	9	10	9
n=40	10	8	9	10
n=60	8	5	5	6
n=80	0	3	3	3
n=100	1	3	0	2
Total	29	28	27	**30**
Class 6				
n=20	10	10	10	10
n=40	5	6	6	9
n=60	10	9	9	6
n=80	10	10	10	10
n=100	2	8	7	8
Total	37	**43**	**42**	**43**

C. MO Optimization Performance

In this section, three commonly used performance metrics, generational distance (GD) [29], spacing (S) and maximum spread (MS) [7], are employed. GD determines the distance of the known Pareto front (PF) from the global Pareto front (GF). Therefore, a lower value in the GD means that all the elements generated are near the global Pareto front. S

measures how evenly the members in the PF are distributed. A value of zero means that all the members in the PF are evenly distributed. MS measures how "well" the GF is covered by the PF. 10 runs are performed for each algorithm on the three selected test cases from Class 2, 3 and 4 respectively to study the robustness and consistency of the algorithms.

1) Problem Class_2_4_14: The test case Class_2_4_14 belongs to Class 2. Class_2_4_14 consists of items which are small since the items are less than a quarter of the size of the bin. As the items are small, there exists many ways to arrange the items in the bin to achieve high stability. Figure 10 summarizes the statistical performance of the different algorithms on 10 runs of the Class_2_4_14 test case. From Figure 10, HMOPSO has the lowest GD which shows the good convergence ability of HMOPSO. MOPSO shows no convergence at all since its GD is very much higher than MOEA and HMOPSO. The MS of MOEA is slightly better than that of HMOPSO, and very much better than MOPSO. In the spacing metric, HMOPSO is the best since it scored the lowest value. From the analysis, it can be seen that the average performance of HMOPSO is the best among the three algorithms.

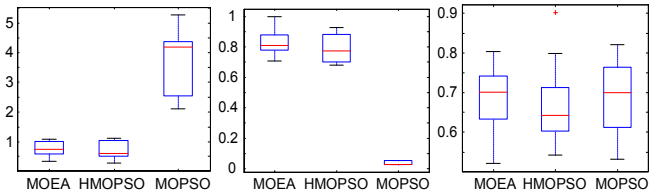


Figure 10: Performance Metric (GD, MS, S) for Class_2_4_14

2) Problem Class_3_4_14: The test case Class_3_4_14 belongs to Class 3 and consists of mainly of small and medium size items. From Figure 11, it can be observed that the Pareto fronts of HMOPSO are nearer to the global Pareto front in most runs as shown by the better GD metric. Again, the MOPSO is stuck at the local minimum. This may be due to the fact that MOPSO is not tailored made to solve the bin packing problem. Although HMOPSO and MOEA have comparable results in the performance metric GD and MS, the average performance of HMOPSO is considered better than MOEA as HMOPSO is better in the Spacing metric.

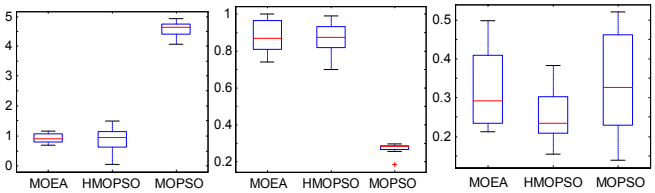


Figure 11: Performance Metric (GD, MS, S) for Class_3_4_14

3) Problem Class_4_4_14: The test case Class_4_4_14 belongs to Class 4 and consists mainly of a mixture of small, medium and large items. This combination of items poses great difficulty for the rearrangement of items in the bin as the large items restrict movement of the other items in the bin. HMOPSO has a slightly better GD than MOEA. However, MOEA has better MS than HMOPSO. This may be due to the fact that the movement of the particle depends mainly on the *pbest* and *gbest*. This makes it difficult to explore new region if there is a break in the Pareto front. This situation may be remedied by introducing a turbulence feature into HMOPSO to make it less likely to be affected by such a situation.

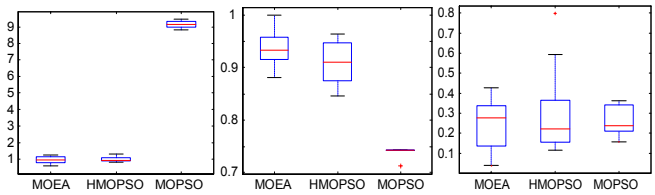


Figure 12: Performance Metric (GD, MS, S) for Class_4_4_14

D. Computational Efficiency

The computation time of the algorithm is very important. If an algorithm takes a long time to converge to the optimal result, then the algorithm is not exactly efficient as it seems to be a brute force attempt.

The three algorithms adopt the same stopping criteria where the simulation will stop after 1000 iterations or when there is no improvement in the last 5 generations. From Figure 13, the simulation time of MOPSO is the shortest for all the three randomly chosen test cases. However, the optimization result obtained by MOPSO is inferior as compared to the result obtained by MOEA and HMOPSO, though the two algorithms take longer time to finish the simulation. This shows that MOPSO may be stuck at the local Pareto front resulting in a short computation time. From the chart, HMOPSO takes lesser time on average to complete a simulation compared to MOEA. It is also observed that the algorithms take a much longer time to compute feasible solutions for Class 2 instances. This is because of the BLF heuristic which takes a longer time to search for free space in the bin to insert a small item. For items of larger size, the BLF heuristic works more efficiently as there are less space to search in a bin to insert an item.

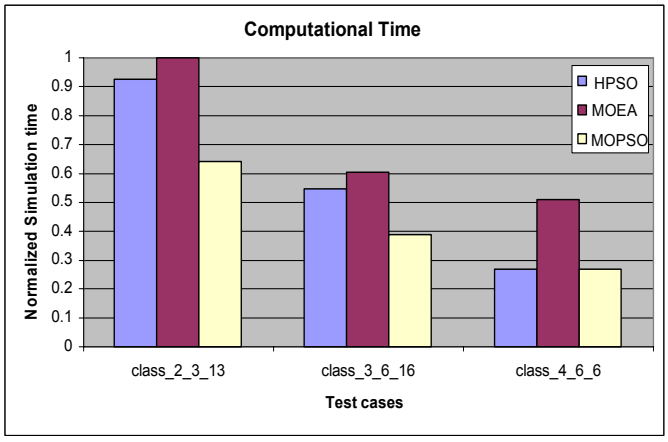


Figure 13: Normalized simulation time for the three algorithms

V. CONCLUSION

In this paper, a mathematical model for MOBPP-2D has been presented and solved by the proposed HMOPSO. The BLF has been chosen as the decoding heuristic since it has the ability to fill in the gaps in the partial layout. The creation of variable length data structure and its specialized mutation operator make HMOPSO a robust search optimization algorithm. It has been shown that HMOPSO performs consistently well with the best average performance on the performance metric, and outperforms MOPSO and MOEA in most of the test cases used in this paper.

REFERENCES

[1] Anily, S., Bramel, J. and Simchi-Levi, D., "Worst-case Analysis of Heuristics for the Bin Packing Problem with General Cost Structures," *Operations Research*, vol. 42, no. 2, pp. 287-298, 1994.

[2] Berkey, J. O. and Wang, P. Y., "Two-Dimensional Finite Bin-Packing Algorithms," *Journal of the Operations Research Society*, vol. 38, pp. 423-429, 1987.

[3] Boutevin, C., Gourgand, M. and Norre, S., "Bin Packing Extensions for Solving an Industrial Line Balancing Problem," *Proceeding of the 5th IEEE Internatinal Symposium on Assembly and Task Planning*, pp. 115-121, 2003.

[4] Brusco, M. J., Thompson, G. M. and Jacobs, L. W., "A Morph-Based Simulated Annealing Heuristic for a Modified Bin-Packing Problem," *Journal of the Operations Research Society*, vol. 48, pp. 433-439, 1997.

[5] Chandra, A. K., Hirschberg, D. S. and Wong, C. K., "Bin Packing with Geometric Constraints in Computer Network Design," *Operation Research*, vol. 26, no. 5, pp. 760-772, 1978.

[6] Dowsland, K. A., "Genetic Algorithms – a Tool for OR," *Journal of the Operations Research Society*, vol. 47, pp. 550-561, 1996.

[7] E. Zitzler K. Deb and L. Thiele. "Comparison of multiobjective evolutionary algorithms: empirical results," *Evolutionary Computation*, vol. 8, no. 2, pp. 173-195, 2000.

[8] Falkenauer, E. and Delchambre, A., "A Genetic Algorithm for Bin Packing and Line Balancing," *IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1186-1192, 1992.

[9] Fonseca, C. M., "Multiobjective Genetic Algorithms with Application to Control Engineering Problems," Dept. Automatic Control and Systems Eng., University of Sheffield, Sheffield, UK, Ph.D. Thesis, 1995.

[10] Fonseca, C. M. and Fleming, P. J., "Multiobjective optimization and multiple constraint handling with evolutionary algorithms – part I: A unified formulation," *IEEE Transactions on Systems, Man and Cybernetics – Part A: Systems and Humans*, vol. 1, no. 28, pp. 26-37, 1998.

[11] Hopper, E. and Turton, B., "A Genetic Algorithm for a 2D Industrial Packing Problem," *Computers & Industrial Engineering*, vol. 37, pp. 375-378, 1999.

[12] Iima, H. and Yakawa, "A New Design of Genetic Algorithm for Bin Packing," *Congress on Evolutionary Computation,* vol. 2, pp. 1044-1049, 2003.

[13] Jakobs, S., "On Genetic Algorithms for the Packing of Polygons," *European Journal of Operational Research*, vol. 88, pp. 165-181, 1996.

[14] Kao, C. Y. and Lin, F. T., "A Stochastic Approach for the One-Dimensional Bin-Packing Problems," *IEEE International Conference on Systems, Man and Cybernetics,* vol. 2, pp. 1545-1551, 1992.

[15] Litvinenko, V. I., Burgher, J. A., Tkachuk, A. A. and Gnatjuk, V. J., "The Application of the Distributed Genetic Algorithm to the Decision of the Packing in Containers Problem," *IEEE International Conference on Artificial Intelligence Systems,* pp. 386-390, 2002.

[16] Liu, D. and Teng, H., "An Improved BL-algorithm for Genetic Algorithm of the Orthogonal Packing of Rectangles," *European Journal of Operational Research*, vol. 112, pp. 413-420, 1999.

[17] Lodi, A., Martello, S. and Vigo, D., "Heuristic algorithms for the Three-Dimensional Bin-Packing Problem," *European Journal of Operational Research*, vol. 141, pp. 410-420, 2002.

[18] Martello, S. and Vigo, D., "Exact Solution of the Two-Dimensional Finite Bin Packing Problem," *Management Science*, vol. 44, pp. 388-399, 1998.

[19] Martello, S. Pisinger, D. and Vigo, D., "The Three Dimensional Bin Packing Problem," *Operations Research*, vol. 48, pp. 256-267, 2000.

[20] Mongeau, M. and Bes, C., "Optimization of aircraft container loading," *IEEE Transactions on Aerospace and Elecronic Systems*, vol. 39, pp. 140-150, 2003.

[21] Pargas, R. P. and Jain, R., "A Parallel Stochastic Optimization Algorithm for Solving 2D Bin Packing Problems," *Proceedings of 9th Conference on Artificial Intelligence for Applications,* pp. 18-25, 1993.

[22] Pimpawat, C. and Chaiyaratana, N., "Using a Co-Operative Co-Evolutionary Genetic Algorithm to Solve a Three-Dimensional Container Loading Problem," *Congress on Evolutionary Computation,* vol. 2, pp. 1197-1204, 2001.

[23] Runarsson, T. P., Jonsson, M. T. and Jensson, P., "Dynamic Dual Bin Packing Using Fuzzy objectives," *Proceedings of IEEE International Conference on Evolutionary Computation,* pp. 219-222, 1996.

[24] Scholl, A. Klein, R. and Jurgens, C., "Bison: A Fast Hybrid Procedure for Exactly Solving the One-Dimensional Bin Packing Problem," *Computers & Operations Research*, vol. 24, Issue 7, pp. 627-645, 1997.

[25] Shigehiro, Y., Koshiyama, S. and Masuda, T., "Stochastic Tabu Search for Rectangle Packing," *IEEE International Conference on Systems, Man, and Cybernetics,* vol. 4, pp. 2753-2758, 2001.

[26] Spillman, R., "Solving Large Knapsack Problems with a Genetic Algorithm," *IEEE International Conference on* Systems, Man and Cybernetics, 'Intelligent Systems for the 21st Century', vol. 1, pp. 632-637, 1995.

[27] Tan, K. C., Lee, T. H., Chew, Y. H., and Lee, L. H., "A hybrid multiobjective evolutionary algorithm for solving truck and trailer vehicle routing problems," *IEEE Congress on Evolutionary Computation*, vol. 3, pp. 2134-2141, 2003.

[28] Van De Vel, H., and Sun, S. J., "An application of the Bin Packing Technique to Job Scheduling on Uniform Processors," *Operation Research*, vol. 42, no. 2, pp. 169-172, 1991.

[29] Van Veldhuizen, D. and Lamont, G. B., "Multi-objective evolutionary algorithm research: A history and analysis", Technical Report TR-98-03, Department of Electrical and Computer Engineering, Air Force Institute of Technology, Ohio, 1998.

[30] Wang, K. P., Huang, L., Zhou C. G. and Pang, W., "Particle Swarm Optimization for Traveling Salesman Problem," *International Conference on Machine Learning and Cybernetics*, vol. 3, pp. 1583-1585, 2003.