Improvement Grouping Genetic Algorithm for Solving the Bin Packing Problem

Yunfei Fan, Jizheng Chu* and Haoran Xu
School of Information Science and Technology, Beijing University of Chemical Technology, Beijing, 100029, China
*chujz@mail.buct.edu.cn

Abstract. An improved grouping genetic algorithm is proposed to solve the one-dimensional bin packing problem. In this algorithm, a new crossover operator is constructed according to the characteristics of the encoding. At the same time, a replacement insertion method is designed to improve the mutation operation. The new mutation operation prevents the algorithm from falling into a local optimum and improves the transmission speed of the best individual in the chromosome. The analysis and comparison of experimental results show that the proposed algorithm can effectively solve the one-dimensional bin packing problem and has reliable global convergence.

1. Introduction
Bin Packing Problem (BBP) is a traditional NP-hard problem, which is widely used in our actual production, life and industrial engineering fields, such as task scheduling, resource management, transportation planning, etc [1]. Therefore, the research on this issue has vital theoretical significance and practical application value.

In terms of computational complexity, since BBP is an NP-hard problem, it is difficult to solve it accurately. So far, people have also proposed a lot of solutions to BBP and conducted extensive research. The main methods of solving the problems are approximate solutions and heuristic algorithms such as genetic algorithms.

From the problem of bin packing to the present, various bin packing algorithms have been proposed continuously. The earliest researched approximate algorithms for BBP are three online algorithms: NF algorithm, FF algorithm, and BF algorithm [2]. In the early 1990s, many scholars invariably considered the advantages and disadvantages of the three algorithms and proposed a class of Bounded-Space online algorithms, including NF-K algorithm, BBF-K algorithm [3], ABF-K algorithm and AFB-K algorithm, their common feature is that the time complexity is close to NF algorithm and the performance is close to FF algorithm. In terms of intelligent optimization algorithms, many scholars have proposed algorithms such as genetic algorithm [4], simulated annealing algorithm [5], and particle swarm algorithm [6] to solve BBP. Intelligent optimization algorithms mainly seek optimization by simulating the evolutionary process or behaviour process of real systems.

2. Problem Formulation
Given a finite set \( O \) (the volume of the item) and two constants, one is \( C \) (the capacity of the bin) and one is \( N \) (the number of bins). Our job is to put as many items as possible into \( N \) bins. Is there a division of the set \( O \) so that the number of bins \( N \) is minimized and the total volume of items in each bin does not exceed \( C \).
If the volume of each bin is specified as 1 \((C = 1)\), then the linear programming method to describe the one-dimensional BBP is:

\[
\begin{align*}
\text{min } Z(y) &= \sum_{i=1}^{n} y_i \\
\text{s.t. } \sum_{i=1}^{n} w_i x_{ij} &\leq Cy_i, i = 1, 2, \cdots, n \\
\sum_{i=1}^{n} x_{ij} &= 1, j = 1, 2, \cdots, n \\
y_i &\in \{0, 1\}, i = 1, 2, \cdots, n \\
x_{ij} &\in \{0, 1\}, i, j = 1, 2, \cdots, n
\end{align*}
\]

(1)

(2)

(3)

(4)

(5)

Where \(w\) is the size of item \(i\), and \(x, y\) are:

\[
\begin{align*}
x_{ij} &= \begin{cases} 1, \text{if item } i \text{ is assigned to bin } k \\ 0, \text{otherwise} \end{cases}, i, j = 1, 2, \cdots, n \\
y_i &= \begin{cases} 1, \text{if bin } k \text{ is used} \\ 0, \text{otherwise} \end{cases}, i = 1, 2, \cdots, n
\end{align*}
\]

(6)

(7)

3. **Improvement grouping genetic algorithm**

The improved grouping genetic algorithm (IGGA) we proposed is an efficient genetic algorithm suitable for grouping problems. The essence of the application to BBP is to divide the items into several groups, and each group corresponds to a bin. Group-based encoding allows different individuals to have different chromosome lengths, which is fitter for solving problems with unknown target bins. Therefore, IGGA uses different encoding methods, and different crossover and mutation strategies based on this encoding method.

3.1. **Encoding**

Falkenaue [7] proved that group-based encoding schemes could avoid encoding from being redundant. The encoding scheme in this paper is also group-based.

The grouping in IGGA consists of two parts: one is the item part and the other is the bin part. The item part is composed of \(n\) genes, where \(n\) is the number of items; the bin part is composed of \(k\) groups of label arrangements, where \(k\) is the number of bins. The value of the item gene takes one of the \(k\) labels in the bin part, indicating that the item gene belongs to the group with the given label. The example of \(n = 9\) and \(k = 5\) is shown in Figure 1.

![Figure 1. Two representations of encoding.](image-url)
There are five bins in the bin part, each represented by ABCDE. The item part indicates that items 3 and 5 are packed in bin A, items 1 and 9 are packed in bin B, items 2 and 8 are packed in bin C, items 4 and 7 are packed in bin D, and items 6 are packed in bin E.

3.2. Fitness function
The fitness function proposed by Falkenauer is used here to evaluate the solution.

\[ F_{IGGA} = \frac{\sum_{i=1}^{n} (F_i / C)^k}{N} \]  

Among them, \( N \) is the total number of bins in the final solution; \( F_i \) is the total volume of the items in the bin \( i \in 1, \cdots, n \); \( C \) is the capacity of the bin; \( k \) is a heuristic index factor, here \( k = 2 \).

3.3. Initial population
The first step of IGGA is to generate an initial population. A good initial population can significantly reduce the execution time and the number of iterations. The items are packed in random order according to the steps of the Best Fit algorithm to generate an initial population.

The core idea of the Best Fit (BF) algorithm is to put the item into the bin with the smallest remaining capacity but larger than the volume of the item every time. If no feasible bin is found, open a new bin.

3.4. Crossover operator
Generally, the crossover operation is to exchange two parent chromosomes with each other in some way to form two new chromosomes. But when we use group encoding, this crossover operator is not a suitable choice. In a GGA, the better the fitness of each group, the better the chromosomes, and the fewer the total number of bins required. So our crossover operator chooses to save those groups with better fitness in order to produce better children. Here we construct the following crossover operation based on encoding characteristics.

1) Each parent is composed of several groups, and each group of the two parents is sorted in descending order of fullness to generate children;
2) Remove the group with duplicate items in the offspring (some non-repeated items in the above group have also been removed);
3) Reinsert the non-repeated items removed in step 2) according to the First Fit Decreasing algorithm.

The core idea of the First Fit Decreasing (FFD) algorithm is to arrange the items in descending order, then traverse all non-empty bins, find the first bin that can put the item, and pack the item in. If no feasible bin is found, open a new bin. Figure 2 is a schematic diagram of a crossover operation.
Suppose there are 9 items \{1, \ldots, 9\} with volumes \{7,5,3,2,5,8,2,6,2\}, and the bin capacity is 10. Randomly give two solutions as parent.

| Fullness | 10 | 9 | 8 | 8 | 5 |
|----------|----|---|---|---|---|
| Items    | A  | B | C | D | E |
| 6.9      | 1.7| 4.8| 3.5| 2 |

| Fullness | 10 | 8 | 8 | 7 | 7 |
|----------|----|---|---|---|---|
| Items    | a  | b | c | d | e |
| 1.3      | 6  | 7.8| 5.9| 2.4|

First father

Second father

Sort in descending order of fullness

Child

Remove group with duplicate items

Reserved group

Idle items

Reinsert

Final

3.5. Mutation operator

Generally speaking, a mutation operator replaces a gene on a certain gene segment in a chromosome code with other alleles of the gene segment to form a new chromosome. The mutation operation often determines the local search ability of the genetic algorithm. In a GGA, the mutation operation generally removes some bins, and then reinserts the items in the removed bins according to the FFD algorithm. However, this operation will reduce the global optimization ability of the algorithm and cause the algorithm to fall into a local optimum.

In view of the above shortcomings, we design a new mutation operation: the replacement insertion method. Following is the description of the replacement insertion method.

1) We traverse each group of chromosomes, sort them in descending order of fullness, and remove the group with low fullness according to the ratio of 0.3 to obtain a number of removed items and record them as idle items.

Figure 2. Illustration of crossover operation.
2) Replace the items in each reserved group with idle items. The principle is to ensure that the total volume of the items in the reserved group will not decrease. In this way, we can develop new chromosomes more comprehensively while retaining the best genes. Figure 3 is a schematic diagram of a mutation operation.

Suppose there are 11 items \{1, ..., 11\} with volumes \{5,4,2,2,7,3,6,5,8,2,6\}, and the bin capacity is 10. Randomly give a chromosome.

| Items | Fullness |
|-------|----------|
| A     | 10       |
| B     | 10       |
| C     | 9        |
| D     | 9        |
| E     | 7        |
| F     | 5        |

Remove some groups by a ratio of 0.3

| Reserved groups | Removed group |
|-----------------|---------------|
| 10, 10, 9, 9    | 7, 5          |
| 3.9, 2.7, 4.5, 6.11 | 1.10, 8       |

Replace

| Items | Final |
|-------|-------|
| A     | 1.8   |
| B     | 2.7   |
| C     | 4.9   |
| D     | 5.6   |
| E     | 3.10,11 |

4. Experimental results
We implemented the algorithm with matlab R2014a and performed experiments on a computer with Intel (R) Core (TM) i5-5200U CPU @ 2.20GHz. Following are the algorithm parameters:

1) Crossover probability: \( P_c = 0.5 \);
2) Mutation probability: \( P_m = 0.33 \);
3) Population size: \( m = 100 \);
4) The maximum number of iterations: \( g = 500 \);
5) Remove ratio in mutation operation: \( P_r = 0.3 \);

Tested in two benchmark problem examples provided by OR-Library. The two class of problem instances are U and T. Class U (u120, u250, u500 and u1000) items are evenly spaced between 20 and 100. Each bin has a capacity of 150 and each group contains 20 instances. Class T (t60, t120, t249 and t501) items have a volume of 25~50, and the capacity of each bin is 100. Each group also contains 20 instances. Class T has three items per bin, so it is more difficult to solve.
In this paper, we compare IGGA with AUGNN, MBS [8], HGGA and MBS ’+ VNS [9]. Table 1 shows the results.

| Problem instance | Class Instances | Number of resolved instances |
|------------------|----------------|-----------------------------|
|                  | AUGNN | MBS | HGGA | MBS’+VNS | IGGA   |
| u120             | 20    | 14  | 12   | 18       | 20     |
| u250             | 20    | 9   | 10   | 18       | 20     |
| u500             | 20    | 3   | 11   | 20       | 20     |
| u1000            | 20    | 0   | 7    | 20       | 20     |
| t60              | 20    | 0   | 0    | 18       | 20     |
| t120             | 20    | 0   | 0    | 20       | 20     |
| t249             | 20    | 0   | 0    | 20       | 20     |
| t501             | 20    | 0   | 0    | 20       | 20     |

The comparison results show that the performance of IGGA is better than any algorithm in the literature, and it solves all instances of U and T, and proves the feasibility and effectiveness of IGGA.

5. Conclusion
We propose an IGGA to solve the one-dimensional BBP. The algorithm has designed a replacement insertion method to develop a new chromosome more comprehensively while retaining the best genes. This method effectively improves the global search ability of the algorithm. The experimental results prove that IGGA has obtained stable and excellent results in a series of benchmark problem examples. In further work, we hope to apply IGGA to higher-dimensional BBP.

Acknowledgments
The work of this paper is supported by the National Natural Science Foundation of China (Grant No. 21676012) and the Fundamental Research Funds for the Central Universities (Project XK1802-4).

References
[1] Ayob, M. , Nazr, M. Z. A. , & Fei, Y. X. . (2013). Local search heuristics for the one dimensional bin packing problems. Journal of Applied Sciences, 13(6), 919-923.
[2] Johnson, D. S. . (1973). Near-optimal bin packing algorithms. Massachusetts Institute of Technology.
[3] Johnson, J. C. S. . (2001). Bounded space on-line bin packing: best is better than first. Algorithmica, 31(2), 115-138.
[4] Lin, Y. T. , & Hou, J. L. . (2012). A genetic algorithm with weight-based encoding for one-dimensional bin packing problem. Applied Mechanics and Materials, 182-183, 2100-2104.
[5] Brusco, M. J. , & Jacobs, G. M. T. W. . (1997). A morph-based simulated annealing heuristic for a modified bin-packing problem. The Journal of the Operational Research Society, 48(4), 433-439.
[6] Qi, Y. , & Wang, J. M. . (2011). The particle swarm optimization algorithm for solving rectangular packing problem. Advanced Materials Research, 186, 479-483.
[7] Falkenauer, E. . (1996). A hybrid grouping genetic algorithm for bin packing. Journal of Heuristics, 2(1), 5-30.
[8] Almeida, R. D. , & Steiner, M. T. A. . (2015). Resolution of 1-D Bin Packing Problem using Augmented Neural Networks and Minimum Bin Slack. 2015 Latin America Congress on Computational Intelligence (LA-CCI). IEEE.
[9] Fleszar, K. , & Hindi, K. S. . (2002). New heuristics for one-dimensional bin-packing. Computers & Operations Research, 29(7), 821-839.