Two-dimensional Bin-packing Problem with Rectangular and Circular Regions Solved by Genetic Algorithm

Nan Wang, Jie-Sheng Wang *, Yong-Xin Zhang, and Tian-Zhu Li

Abstract—**Bin-packing Problem (BPP) is to put a certain number of items into a fixed size container and strive to achieve a certain optimal goal under certain constraints. Aimed at simulating common plate cutting problems based on rectangles and circles, the mathematical model of a two-dimensional bin-packing problem was established by considering the non-superposition of items and the constraint of the box in the form of coordinates. Then a method based on Genetic Algorithm (GA) was proposed to solve this two-dimensional BPP with rectangular and circular areas. The GA parameters are optimized from the single change of mutation probability and the unified change of mutation probability. Simulation results verify the effectiveness of the proposed algorithm for solving two-dimensional bin-packing problems with rectangular and circular regions.**

Index Terms—**Bin-packing Problem； Genetic algorithm； Performance comparison**

I. Introduction

Bin-packing problem is a classical combination optimization problem. The so-called combination optimization problem refers to the process of finding the optimal solution from a finite set of feasible solutions [1]. When solving this kind of optimization problems, the time complexity and space complexity of the algorithm are the main constraints. According to the solving difficulty, the problems are divided into P (Polynomial), NP (Non-deterministic Polynomial) and NPC (Non-deterministic Polynomial Complete). The bin-packing problem is characterized by discontinuity, constraint, non-differential and non-linearity in the process of solving the problem, which is attributed as NP- hard problem in academic circles.

Generally, BPP can be divided into one-dimensional bin-packing problem, two-dimensional bin-packing problem and three-dimensional bin-packing problem according to the space where the object belongs. The one-dimensional BPP is restricted by a specific factor. The two-dimensional BPP is limited to two factors. In two-dimensional space, it is mainly the form of restricting length and width. The three-dimensional BPP is mainly limited to length, width and height. Daily life is reflected in cargo loading such as containers, aircraft cabins, etc. [2-4]. The optimization goal of the BPP is to ensure that the contents of the box do not exceed its own capacity, and strive to use the least amount of boxes to load all the goods, which is reflected in the loading of containers, aircraft cabins, etc. In fact, in the face of the combination and arrangement of graphics in a limited area, container loading in logistics industry, human resource allocation in management industry, plate cutting in industrial production and so on, can be modeled and solved by bin-packing problem. In practical applications, such as plate cutting problem in industrial field, circuit board design problem, multiprocessing task scheduling, resource allocation, file allocation, memory management and other underlying operations in the field of computer science are practical applications of bin-packing problem [5-9].

Bin-packing problem belongs to NP-HARD class problem, and it is difficult to solve it accurately. At present, a variety of approximate algorithms have been proposed to solve the one-dimensional packing problem, such as NFA, FFA, BFA, etc. [10-14]. The online and offline algorithms of general heuristic algorithms and the intelligent optimization algorithms was proposed to solve two-dimensional bin packing problem, such as genetic algorithm (GA), simulated algorithm (SA) and Tabu search (TS) [15-17]. This paper mainly studies the two-dimensional bin-packing problem with rectangular and circular areas based on GA, and verifies the effectiveness of the proposed method through simulation experiments.

II. Mathematical Model of Two-dimensional Bin-packing Problem and Its Solution Method

A. Mathematical Model of Two-dimensional Bin-packing Problem

The two-dimensional bin-packing problem focuses on two dimensions, so the two-dimensional bin-packing problem is usually divided into 2DBP (2-Dimensional Bin-packing Problem) and 2DSP (2-Dimensional Strip-packing Problem). 2DBP refers to that when the width and height of the box are determined, the rectangular objects with uncertain length and width are put into the box, and the target is to minimize the number of boxes，which is widely used in the cargo loading and containers. 2DSP means that the rectangle with indefinite

Manuscript received October 8, 2020; revised December 30, 2020. This work was supported by the National Natural Science Foundation of China (Grant No. 71772082 and 71472080).

Nan Wang is a doctoral candidate of College of Economics and Management, Nanjing University of Aeronautics and Astronautics, Nanjing, 211106, P. R. China. (e-mail: nanwkuhu@163.com).

Jie-Sheng Wang is a professor of School of Electronic and Information Engineering, University of Science and Technology Liaoning, Anshan, 114051, P. R. China. (Corresponding author, phone: 86-0412-2538246; fax: 86-0412-2538244; e-mail: wang_jiesheng@126.com).

Yong-Xin Zhang is an undergraduate of School of Electronic and Information Engineering, University of Science and Technology Liaoning, Anshan, 114051, P. R. China. (e-mail: 1039924966@qq.com).

Tian-Zhu Li is a professor of School of Business Administration, University of Science and Technology Liaoning, Anshan, 114051, P. R. China. (e-mail: ltzuestc@163.com).

length can be planned out on the strip board with fixed width and infinite length, and the objective is to minimize the length of strip plates. This modeling is widely used in the field of plate cutting and so on [14].

In the process of mathematical modeling of BPP, the non-superposition between the items and the further constraint of the box in the form of coordinates are added to make it closer to the actual problem. The two-dimensional BPP can be described as follows. $P = (P_1,P_2,\cdots,P_n)$, $0 < P_i \leq 1$, and $Q(P_i)$ is the area of the i-th item, $1 \leq i \leq n$; The area of the box is set to 2; How to try to put $P_1,P_2,\cdots,P_n$ into as many boxes as possible is the solved problem. In the simulation experiments, the lower left corner of the box is taken as the coordinate origin, the transverse direction is x axis and the longitudinal direction is Y axis. Let (x_{i1},y_{i1}) be the coordinates of the lower left corner of the box, and (x_{i2},y_{i2}) be the upper right coordinate of the box. The mathematical model of the bin-packing problem is defined as follows:

$$minz(y) = \sum_{i=1}^{n} R_i \tag{1}$$

$$s.t \sum_{j=1}^{n} Q(P_i)\, S_i < R_i \ , i \in N = (1,2,\cdots,n) \tag{2}$$

$$x_{i1} > x_{i2},\ y_{i1} > y_{i2} \quad i \in N \tag{3}$$

$$\sum_{j=1}^{n} d_i = 1 \quad j \in N \tag{4}$$

$$\sum_{i=1}^{n} S_i = 1 i \in N \tag{5}$$

where, $R_i = 0$ or 1 $i \epsilon N$, $S_i = 0$ or 1 $i \in N$; $y_i = 1$ means that box i is put into the item, otherwise it means that box i is empty. $d_j = 1$ means that the item itself does not appear to be superimposed, otherwise it means there is superposition effect; $S_i = 1$ indicates that item i is put into the box, otherwise, it means that item i is not put into the box.

B. Solution Methods of Bin-packing Problems

The bin-packing problem widely exists in the fields of cutting of industrial plates, loading of containers and other goods in logistics transportation, and typesetting of circuit board devices. But there is no accurate algorithm to solve this kind of NP problem in effective time. According to the characteristics of the algorithm for solving the bin-packing problem, it can be divided into deterministic algorithm, general heuristic algorithm and modern heuristic algorithm. The deterministic algorithm for bin-packing problem is to find the optimal solution or get the optimal packing method through finite iterations or layout attempts. It mainly forms enumeration algorithm, branch and bound algorithm and branch pricing algorithm. However, the ability of deterministic algorithms to solve the bin-packing problem increases rapidly with the increase of the control parameters or scale of the problem. Therefore, in order to maximize the efficiency of the solution, many heuristic algorithms were proposed to solve it. The general heuristic algorithms based on hierarchical strategy mainly include Next-Fit Decreasing Height strategy (NFDH), First-Fit Decreasing Height strategy (FFDH) and Best-Fit Decreasing Height strategy(BFDH). These three common hierarchical algorithms are shown in Fig. 1. Coffman et al. proved that if the height of a rectangular object is standardized, that is to say that the maximum height is 1, the time complexity satisfies:

$$NFDH(I) \leq 2 * OPT(I) + 1 \tag{6}$$

$$\mathrm{FFDH}(I) \leq \frac{17}{10} * OPT(I) + 1 \tag{7}$$

The time complexity of both algorithms is O ($nlogn$). From the actual effect, although three algorithms of NFDH, FFDH and BFDH have been optimized to different degrees, there are still many shortcomings from the actual renderings and there is still a lot of remaining area. Based on the above characteristics, the non-hierarchical algorithms were proposed. The most classic algorithms are BL (Bottom and Left) algorithm and FF (Fall Free) algorithm. The packing process is shown in Fig. 2 (the number in this figure means the packing sequence). Modern heuristic algorithms are a further development of general heuristic algorithms. They are mainly based on Genetic Algorithm (GA), Simulated Algorithm (SA), Tabu Search (TS) and other intelligent optimization algorithms for solving these kinds of combination optimization problems (NP hard problems). This paper mainly studies the two-dimensional bin-packing problem with rectangular and circular areas based on Genetic Algorithm [18-19].

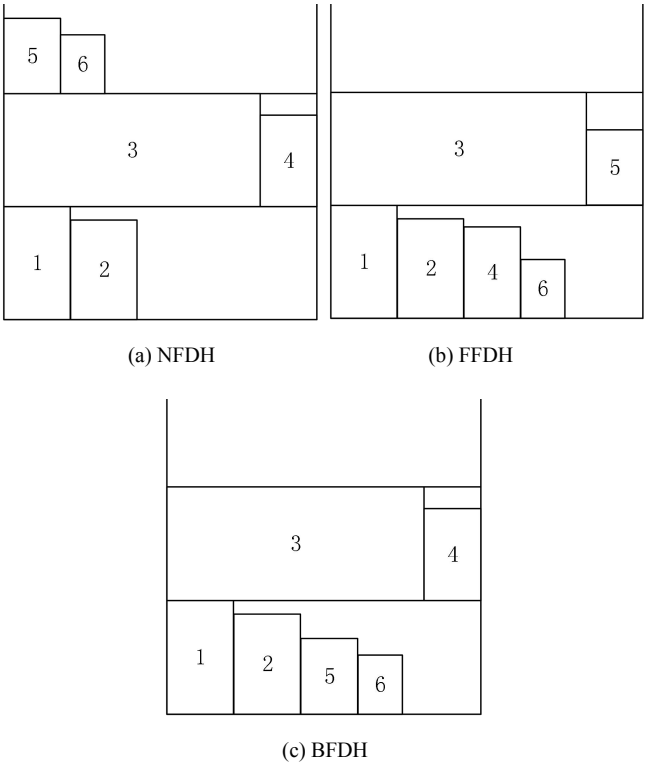
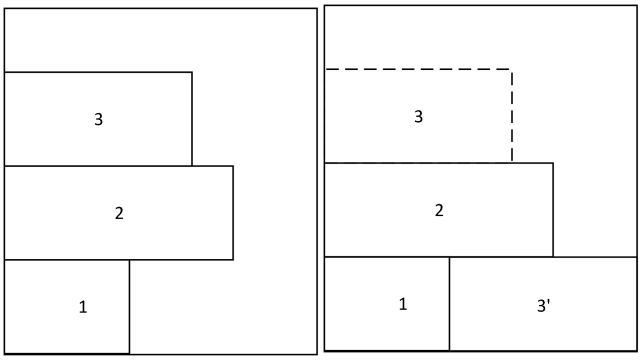


Fig. 1 Schematic diagram of three common hierarchical algorithms.



(a) BL algorithm (b) FF algorithm

Fig. 2 Comparison of BLA and FFA operation process.

III. Solving Two-dimensional Bin-packing Problems with Rectangular and Circular Regions Based on Genetic Algorithms

A. Overview of Genetic Algorithms

Genetic Algorithm (GA) is a swarm intelligence optimization algorithm proposed by Professor J. Holland in 1975 based on the genetic and mutation process in Darwin's theory of biological evolution. In genetics, organisms continuously propagate and evolve based on genes in chromosomes. When reproducing a new generation, chromosomes are used as carriers to proceed the select, crossover, and mutate operators to form new chromosomes, and then new individuals are generated. New individuals are screened by the principle of "survival of the fittest" in nature, and individuals suitable for the environment are retained, propagated and evolved again and again. GA has two significant features which are mainly global search features and implicit parallelism. The genetic operators are defined to act on the population $G(t)$. In order to obtain a new generation of population $G(t+1)$, the following genetic operations will be carried out. The flowchart of GA is shown in Fig. 3.

1) Selection operator. Through the specific "survival of the fittest", the corresponding fitness rate of each individual is calculated, and the population with the highest "survival rate" is selected. At this time, some of the selected t-th generation population $G(t)$ are close to the optimal solution, that is to say that the excellent individuals are inherited to the $(t+1)$ generation population $G(t+1)$.

2) Crossover operator. In the t-th generation population $G(t)$, two parents are selected to set up gene exchange operation (according to certain crossover probability) to form new individuals. Operate according to a certain probability, that is, set the crossover probability.

3) Mutation operator. For the t-th generation population $G(t)$, the process of a gene of each individual is changed by mutation probability to form a new individual.

B. Solving Two-dimensional Bin-packing Problems with Rectangular and Circular Regions Based on Genetic Algorithm

1) Coding Strategy

Based on the characteristics of the two-dimensional bin-packing problem, the area of the figure is initially set, and the side length *aa* and *bb* of the rectangle are set in turn to form the basic figure with the area of *aa* * *bb*. The diameter of the circle is set to cc, so that its constituent area is cc*cc*pi/4.

aa
$=$ [0.1 0.6 0.5 0.2 0.1 0.2 0.4 0.4 0.3 0.4 0.25 0.75 0.3 0.1 0.2 0.35 0.25 0.4 0.1 0.2 0.2 0.3 0.1 0.4 0.25 0.25 0.3 0.2 0.5 0.7];

bb
$=$ [0.1 0.2 0.1 0.2 0.1 0.2 0.1 0.2 0.1 0.4 0.25 0.4 0.25 0.4 0.25 0.4 0.25 0.3 0.2 0.4 0.3 0.35 0.5 0.6 0.5 0.75 0.7 0.2 0.3 0.3];

cc
$=$ [0.1 0.5 0.75 0.7 0.2 0.3 0.3 0.4 0.25 0.4 0.25 0.2 0.1 0.2 0.4 0.25 0.1 0.2 0.1 0.2 0.1 0.4 0.25 0.3 0.2 0.4 0.3 0.35 0.5 0.6];

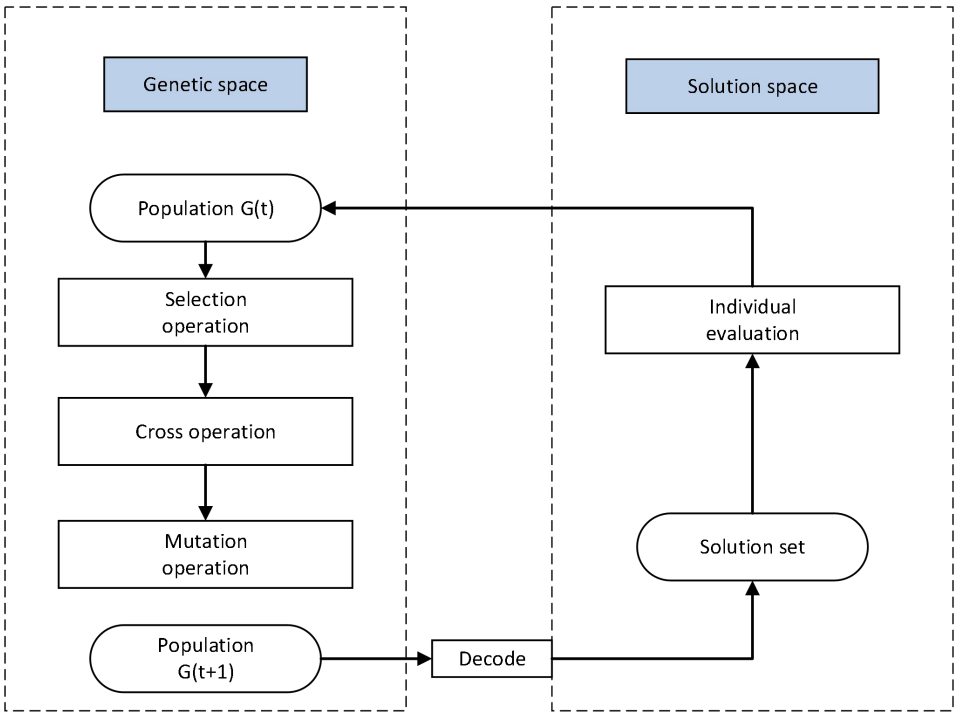


Fig. 3 Flowchart of genetic algorithms.

In this paper, the combination of binary and decimal coding form is adopted in the coding strategy, and each individual exists in the form of column, and the whole row is defined separately. It is mainly considered from the following two aspects. On the one hand, it can improve the application efficiency; on the other hand, it can define enough individuals. In this paper, binary algorithm is used in the first and second lines of coding. When the first line is 1, it means that it can be displayed, and if it is 0, it cannot be displayed. Eq. (8) is used to randomly generate 0 and 1 matrices satisfying normal distribution. In the second line, when it is 1, it means to rotate 90° and if it is 0, it means to keep unchanged for packing. Eq. (9) is a matrix satisfying the normal distribution between 0 and 1 (rotation is not involved in the program with circle as the model). The third and fourth lines represent the (x,y) coordinates of the graph. When to form the initial coordinates (x,y), they are generated by adding and subtracting calculations based on m^2 (the lowest value of side length), which are shown in Eq. (10) and (11).

$$G1(1,:) = double(rand(1,Nb) < 0.2) \tag{8}$$

$$G1(1,:) = double(rand(1,Nb) < 0.5) \tag{9}$$

$$G1(3,:) = m2 + (a - m2) * rand(1,Nb) \tag{10}$$

$$G1(4,:) = m2 + (b - m2) * rand(1,Nb) \tag{11}$$

where, a and b refer to the side length of the box.

2) Fitness Function and Penalty Function

In the process of practical optimization, there are cases that graphics are superimposed and beyond the box. For this reason, the actual comparisons are made based on coordinates. In the process of generating the rectangle, based on the initial coordinates, add and subtract 1 / 2 of the side length to form the lower left coordinate (x_1,y_1) and the upper right coordinate (x_2,y_2). Here x_1,y_1 are compared with 0 so as to select the larger value, and x_2,y_2 are to select the minimum value by comparing with the length and width of the box. In order to ensure that the actual coordinate points are in the box, x_1 and y_1 are always kept at the minimum during this process. If they are violated, they cannot be displayed in the box.

In addition, $(aaa/2 <= x(n))\&\&(x(n) \le a - \frac{aaa}{2})\&\&(bbb/2 <= y(n))\&\&(y(n) <= b - bbb/2)$ is calculated for comparison. If it is true, it must be completely in the box, otherwise the penalty accumulation will be carried out through the penalty function show in Eq. (14). In order to ensure that there is no superposition effect between two items, the distance between the basic coordinates between the two items is calculated and compared with half of the side length of the two items. If it is less than, the superposition effect will be generated, and the penalty functions shown in Eq. (15)-(16) will be carried out.

$$penalty = 0.2 * a * b \tag{12}$$

$$nac = 0.8 \tag{13}$$

$$fitness = A - nac * (A0 - A) - \text{penalty} \tag{14}$$

$$fitness = fitness - 2 * nac * Ac \tag{15}$$

$$fitness = fitness - \text{penalty} \tag{16}$$

$$[\text{fb}, \text{bi}] = \max(fitnessss) \tag{17}$$

Eq. (12) and (13) is adopted to set the initial values of the penalty function. Eq. (12) to (16) is used to carry out the penalty functions under different situations, and finally select the maximum individual fitness through Eq. (17). Here $fitnesses$ means individual fitness, nac means negative area coefficient, $penalty$ means penalty constant, A means figure area, A0 means actual area of each figure, AC means superimposed area, and [fb, bi] is the index position of the individual with largest area.

3) Selection Operator and Crossover Operator

Before the crossover operation is performed, the proportion operator is clarified. In this paper, the Roulette Wheel selection method is used to select and generate the parent chromosome so that it can be used in the crossover and mutation operators. In the crossover operation, the two successive parents were crossed, and the visual individual, the rotation of the individual and the setting of individual position gene were performed respectively. Here the $rand$ function is used to generate random numbers for crossover probability. In the process of algorithm implementation, when the generated random number is less than 0.5, there is no rotation and visual crossover operation. When the random number is greater than 0.5, whether the rotation and visualization of the offspring are exchanged, and the operation of the offspring is the opposite. When the position gene is operated, the function $i3 = 1 + ceil(2 * rand)$ is set to generate 1, 2 and 3 for random operation. When the generated value by switch function is 1, the offspring (x,y) is 1 / 2 of the sum of the parents' genes. When the generation value is 2, the offspring do not perform crossover operation. When the generation value is 3, the offspring genes are exchanged.

5) Mutation Operator

There are four main types of operations in the mutation operator, and mutation operations are performed on the encoding. The mutation operations are visual mutation operation, rotation mutation operation, large Gaussian mutation operation, and small Gaussian mutation operation. The random numbers generated by $rand$ function is used to judge whether the mutation operation is carried out. At the beginning, a fixed mutation probability value is initialized. When the random number is less than the mutation probability, the mutation operation is carried out. This article mainly carried out four mutation operations on the gene positions. The visual mutation operation is similar to the rotation mutation operation. The following function is used for the mutation operation.

$$ir = ceil(Nb * rand); G1(2,ir) = double(rand < 0.5);$$

Respectively, a certain position of the position sequence is exchanged through two randomly generated position nodes.

By using the $randn$ function in MATLAB, the random number satisfying the normal distribution is randomly generated and multiplied by a constant coefficient and added to a certain position of the position sequence. It may be mainly divided into two ways: by multiplying the larger interval between $[0.01,0.1)$ and the smaller interval between $[0.001,0.01)$ to adjust accordingly. By mutating whether the sequence is rotated; by mutating whether it is visible. Here we use the following equations to make a position variation (for example, a large Gaussian jump is performed on a certain position, and a small Gaussian jump can change the coefficient from 0.05 to 0.005).

$$ir = ceil(Nb * rand);$$
$$G1(3{:}4,ir) + [0.05 * a * randn;\ 0.05 * b * randn];$$

The entire genetic algorithm optimization process is designed through for loop programming, where for loop is used to optimize each individual.

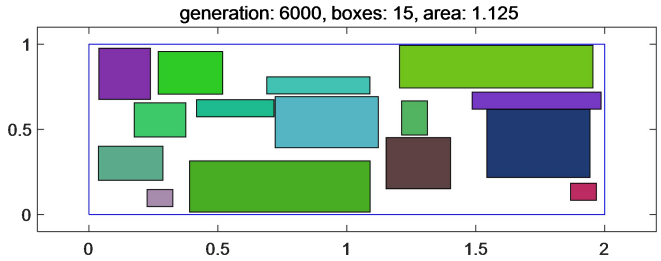
IV. Simulation Experiments and Result Analysis

The number of items in the two-dimensional bin-packing problem solved in this paper is 30. The length and width of the rectangular box or the radius of the round box are determined. The area is 1*2. The population size in the genetic algorithm is 500 and the number of iterations is 6000. In this paper, the simulation experiments and result analysis are carried out by changing position mutation probability and comprehensively changing mutation probability.

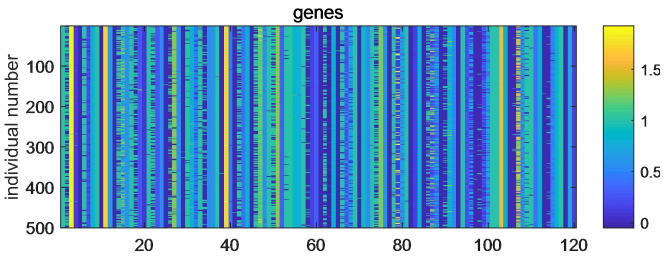
A. *Solving Two-dimensional Bin-packing Problems with Rectangular Area Based on Genetic Algorithm*

1) Single Change Position Mutation Probability

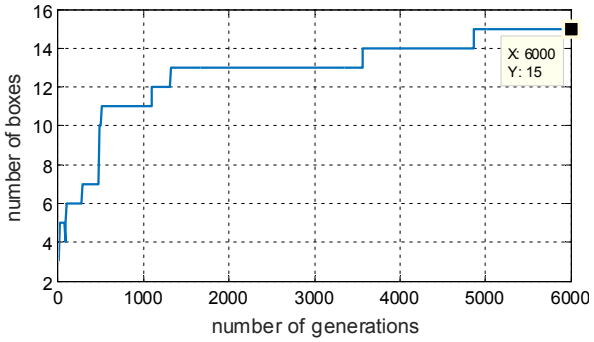
From the perspective of biological heredity, the probability of mutation occurring in the process of biological evolution and development is also very small. Therefore, the probability of large and small Gaussian jumps is set here to be 0.01 and 0.02 respectively. The probability of random rotation and random visibility (that is, put into the box) is 0.05. The simulation results are shown in Fig. 4-6. It can be seen form Fig. 4 that when the probability of position mutation operator is 0.01, the number of iterations continues to 6000 times. Combined with Fig. 4(c)-(d), it is known that the utilization area is continuously increased by iterative optimization. When the number of control graphics changes, the first number of loaded objects is 3. Afterwards, through continuous optimization, when the number of items is 5, through continuous optimization, optimization is carried out to reduce the number of items and increase the utilization area. In this simulation, with the increase of the number of iterations, the space utilization rate has been significantly improved. In the simulation results, the optimization changes are mainly concentrated in the number of iterations within 1000. According to Fig. 4(a), when the final iteration number is 6000, a total of 15 items are loaded, and the area utilization rate is 56.25%. It can be seen from Fig. 5 that when the probability of position mutation operator is 0.05, the number of iterations continues to 6000 times. Combined with Fig. 5(c)-(d), it is known that through iterative optimization, the utilization area is continuously increased, and the number of loaded items is 7. After optimization, when the number of loaded items is 3, the space utilization rate has been significantly improved, which clearly reflects the results to minimize the number of loaded times and maximize the area utilization. According to Fig. 5(a), when the number of iterations is 6000, a total of 9 times are loaded, and the area utilization rate is 69.88%. It can be seen from Fig. 6 that when the probability of the position mutation operator is 0.1, the number of iterations lasts until 6000 times. According to Fig. 6(c)-(d), the optimization process is mainly concentrated in the number of iterations within 3000, and the main optimization process is concentrated in the number of boxes with 7 and 8 items. After multiple optimization calculation, the space utilization rate has been greatly improved. According to Fig. 6(a), when the final iteration number is 6000, a total of 8 items are loaded, and the area utilization rate is 64.13%.



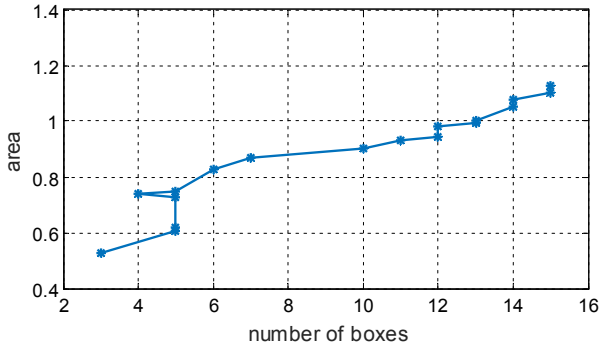
(a) Simulation diagram



(b) Gene diagram

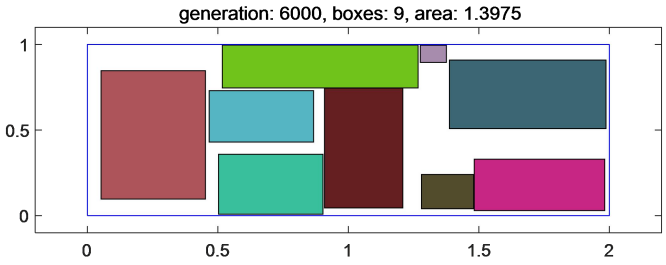


(c) Coordinate diagram of number of iterations and boxes

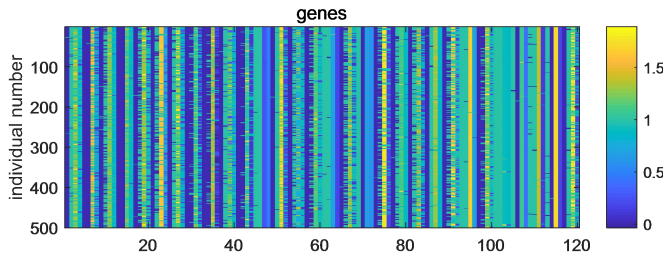


(d) Coordinate diagram of cumulative figure area and number of boxes

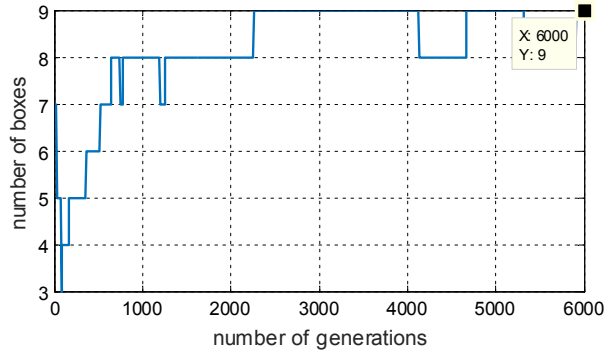
Fig. 4 Simulation results when the control position variation probability is 0.01.



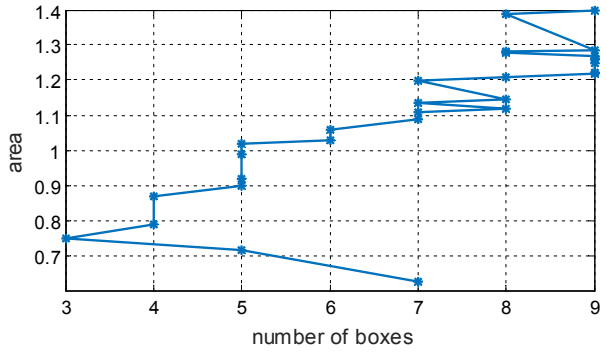
(a) Simulation diagram



(b) Gene diagram

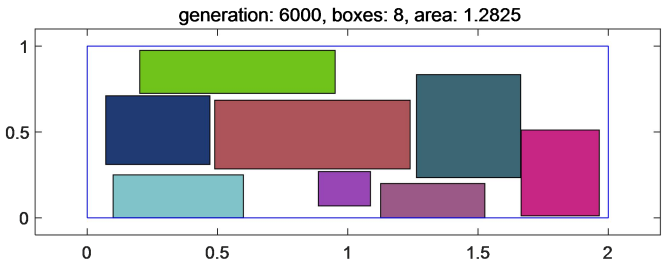


(c) Coordinate diagram of number of iterations and boxes

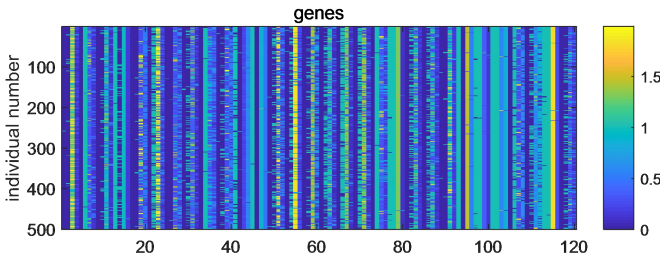


(d) Coordinate diagram of cumulative figure area and number of boxes

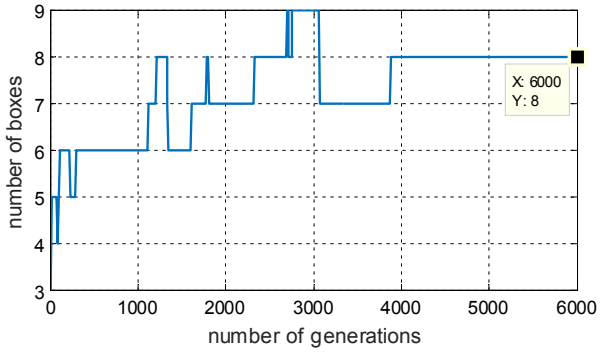
Fig. 5 Simulation results when the probability of control position variation is 0.05.



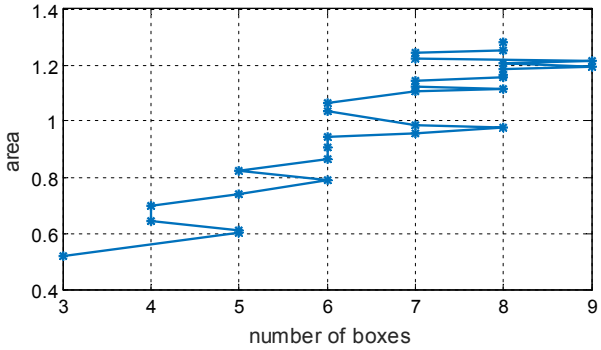
(a) Simulation diagram



(b) Gene diagram



(c) Coordinate diagram of number of iterations and boxes

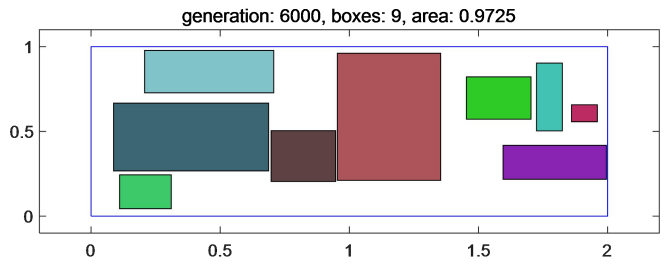


(d) Coordinate diagram of cumulative figure area and number of boxes

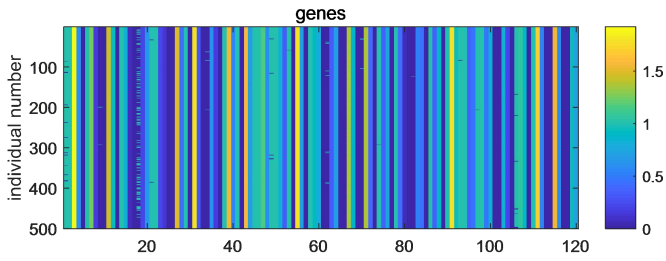
Fig. 6 Simulation results when the probability of control position variation is 0.1.

2) Uniform Change Mutation Operator

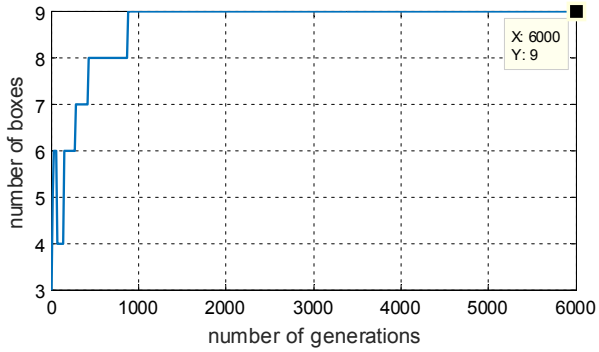
The simulation results after uniformly changing the probability of mutation operators are shown in Fig. 7-8. It can be seen from Fig. 7 that when the probability of uniform mutation operator is 0.01, the number of iterations continues to 6000. According to Fig. 7(c)-(d), the main optimization stage mainly occurs in the number of iterations less than 1000. For nodes with 6 loaded items, it is obvious that the optimization process has changed. When the lower area is the same, the scheme with less loaded items is preferred. According to Fig. 7(a), when the number of iterations reaches 6000, a total of 9 times are loaded, and the area utilization rate is 48.63%. It can be seen from Fig. 8 that when the probability of uniform mutation operator is 0.05, the number of iterations continues to 6000. It can be seen from Fig. 8(c)-(d) that the main optimization stage mainly occurs in the number of iterations less than 1000. In this simulation process, the number starting point of loading items is 6. In the process of continuous optimization, the optimization effect of the number of loaded items is not very obvious. According to Fig. 8(a), when the number of iterations reaches 6000, a total of 11 items are loaded, and the area utilization rate is 56.5%.



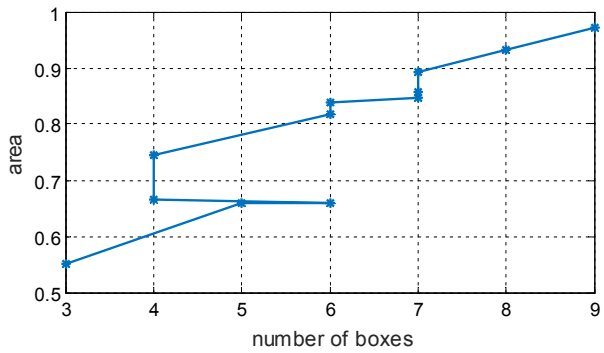
(a) Simulation diagram



(b) Gene diagram

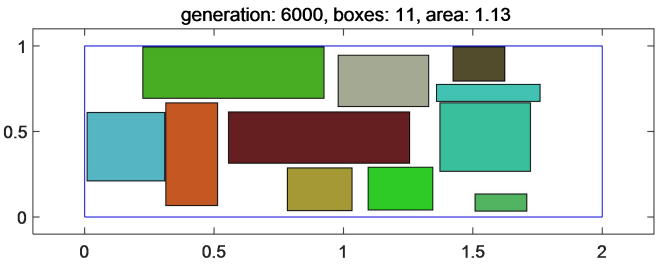


(c) Coordinate diagram of number of iterations and boxes

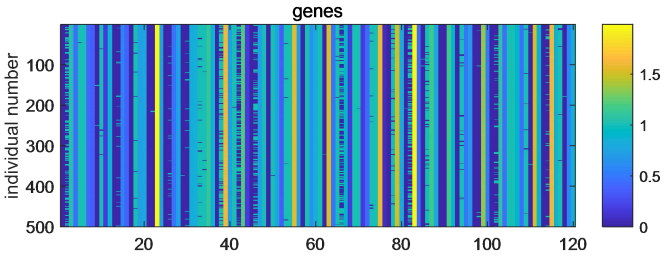


(d) Coordinate diagram of cumulative figure area and number of boxes

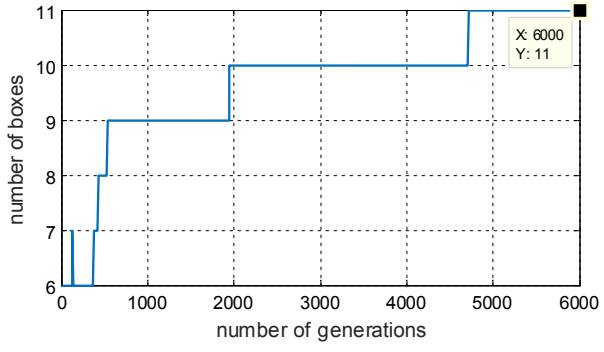
Fig.7 Simulation results with uniform mutation operator of 0.01.



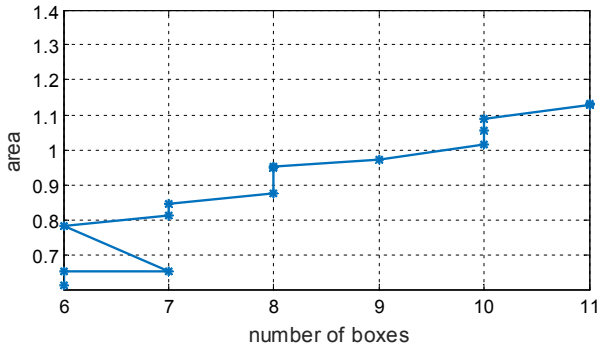
(a) Simulation diagram



(b) Gene diagram



(c) Coordinate diagram of number of iterations and boxes



(d) Coordinate diagram of cumulative figure area and number of boxes

Fig. 8 Simulation results with uniform mutation operator of 0.05.

TABLE 1 OPTIMIZATION PARAMETERS AND PERFORMANCE COMPARISON

Mutation operator	Position mutation operator	Visual mutation operator	Rotational mutation operator	Large Gaussian mutation operator	Small Gaussian mutation operator	Number of boxes	Utilization rate	Number of iterations
Single control	0.01	0.05	0.05	0.01	0.02	15	56.25%	5990
	0.05	0.05	0.05	0.01	0.02	9	69.88%	6000
	0.1	0.05	0.05	0.01	0.02	8	64.13%	6000
Unified control	0.01	0.01	0.01	0.01	0.01	9	48.63%	6000
	0.05	0.05	0.05	0.05	0.05	11	56.5%	6000

The performance comparison between single change location mutation probability and unified change mutation probability is shown in Table 1. Through the simulation, it is found that when the probability of mutation operator is changed alone to 0.05, the area utilization ratio is relatively small peak. Traditional optimization mainly focuses on the number of iterations within 1000. With the superposition of the number of iterations, the number of relative optimization is less.

When the probability of mutation operators is changed uniformly, compared with the single change mutation operator probability, the occurrence of mutation is limited in some way, and the mutation probability is greatly reduced compared with the decentralized control mutation. From a comprehensive perspective of the two sets of controls strategy, when the number of iterations exceeds 1000, the decentralized control variation optimization effect is more obvious.

B. *Solving Two-dimensional Bin-packing Problems with Circular Area Based on Genetic Algorithm*

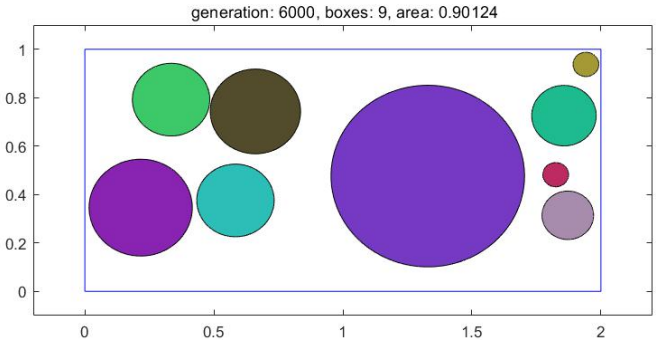
1) Single Change Position Mutation Probability

From the perspective of biological genetics, the probability of mutation in the process of biological evolution and development is also very small. Therefore, the large and small Gaussian jump probability are set as 0.01 and 0.02 respectively, and the probability of random rotation and random visibility (that is, put in the box) is 0.05. The simulation results are shown in Fig. 9-11.

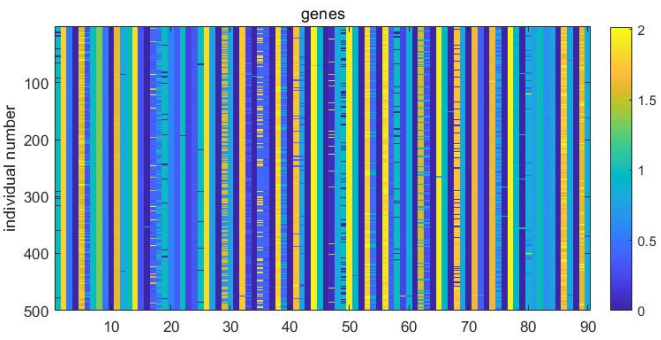
It can be seen from Fig. 9 that when the probability of position mutation operator is 0.01, the number of iterations lasts until 6000 times. According to Fig. 9(c)-(d), it is found that when the number of items is 3 and 4, the number of loaded items is optimized several times between 3 and 4. In the process of continuous optimization, the area utilization rate is gradually improved. According to Fig. 9(a), when the final number of iterations is 4870, a total of 9 items are loaded, and the area utilization rate is 45.06%.

It can be seen from Fig. 10, when the probability of position mutation operator is 0.05, the number of iterations continues to 6000 times. According to Fig. 10(c)-(d), it is known that the utilization area is continuously increased through iterative optimization, and the main optimization process is concentrated on loading nodes with 4 and 5 items. In the last optimization process, although the number of loaded items is not changed, the utilization rate of area is obviously improved. According to Fig. 10(a), when the number of iterations is 4990, a total of 9 items are loaded, and the area utilization rate is 55.96%.

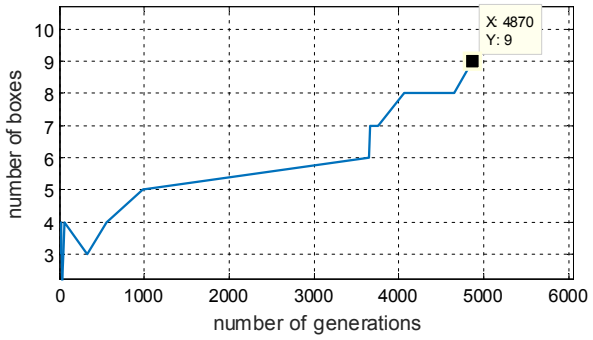
It can be seen from Fig. 11 that when the probability of position mutation operator is 0.1, the number of iterations lasts until 6000 times. According to Fig. 11(c)-(d), 5 items are loaded four times. After optimization, it is reduced to two items. However, according to Fig. 11(a), when the final iteration number is 5930, a total of 9 items are loaded, and the area utilization rate is 53.21%.



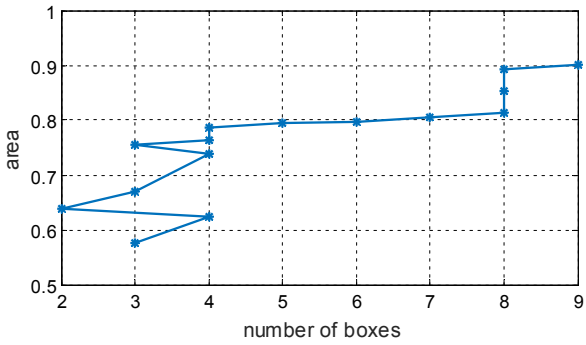
(a) Simulation diagram



(b) Gene diagram

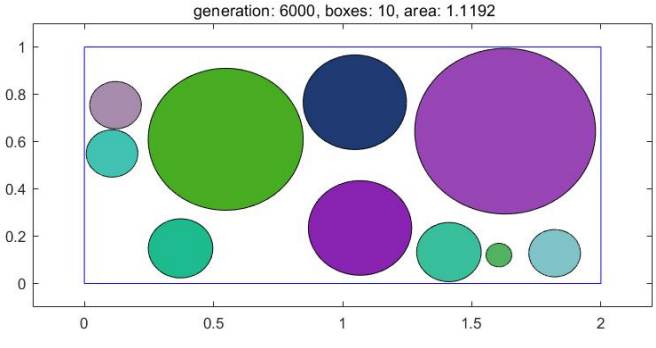


(c) Coordinate diagram of number of iterations and boxes

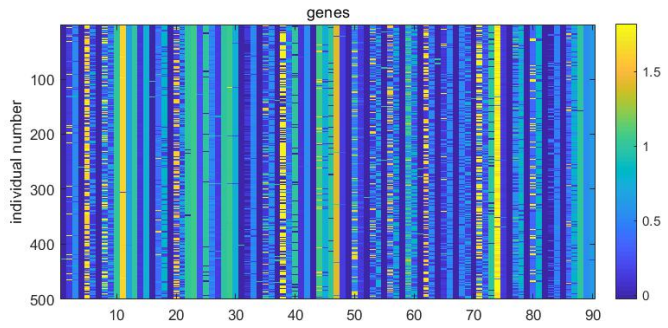


(d) Coordinate diagram of cumulative figure area and number of boxes

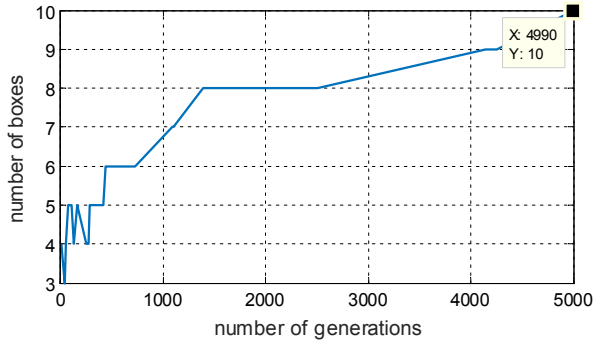
Fig. 9 Simulation results when the control position variation probability is 0.01.



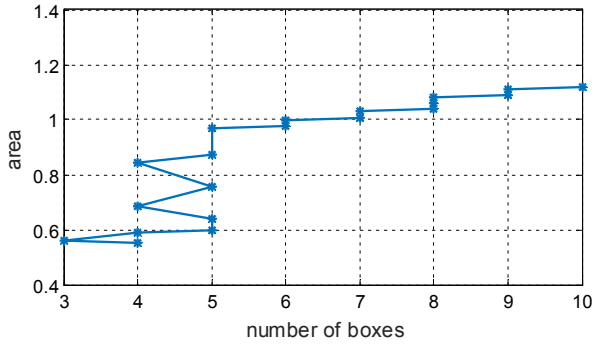
(a) Simulation diagram



(b) Gene diagram

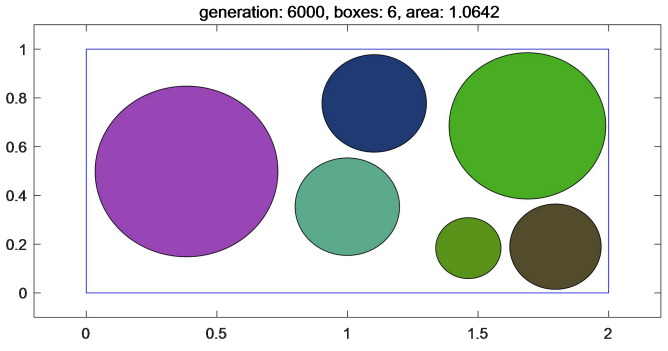


(c) Coordinate diagram of number of iterations and boxes

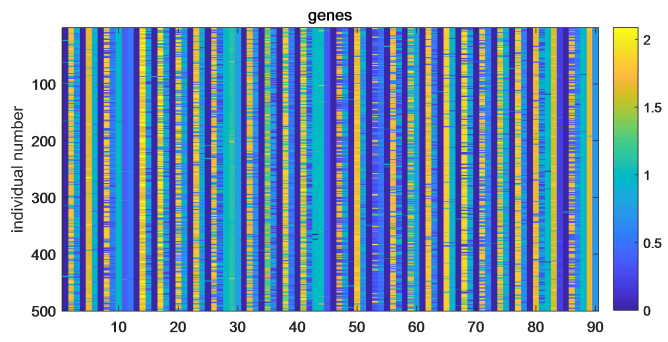


(d) Coordinate diagram of cumulative figure area and number of boxes

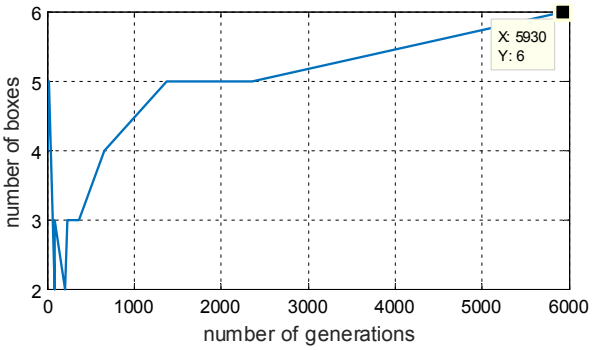
Fig. 10 Simulation results when the probability of control position variation is 0.05.



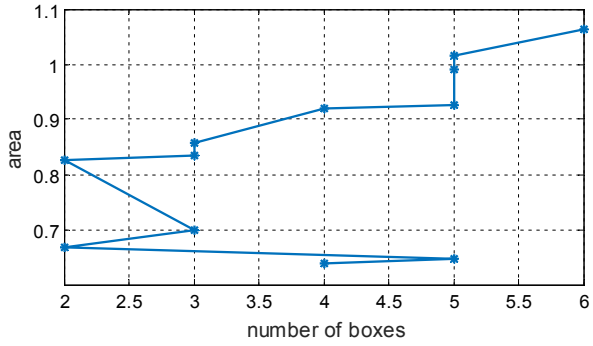
(a) Simulation diagram



(b) Gene diagram



(c) Coordinate diagram of number of iterations and boxes



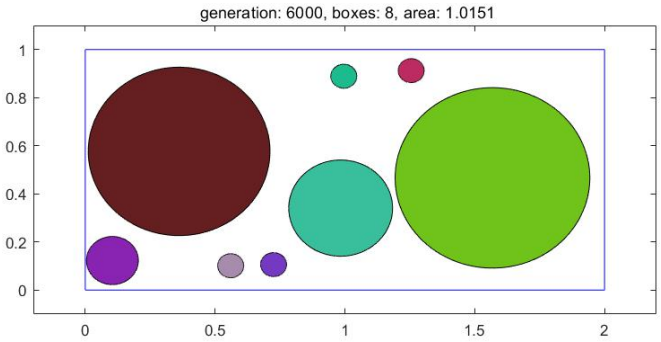
(d) Coordinate diagram of cumulative figure area and number of boxes

Fig. 11 Simulation results when the probability of control position variation is 0.1.

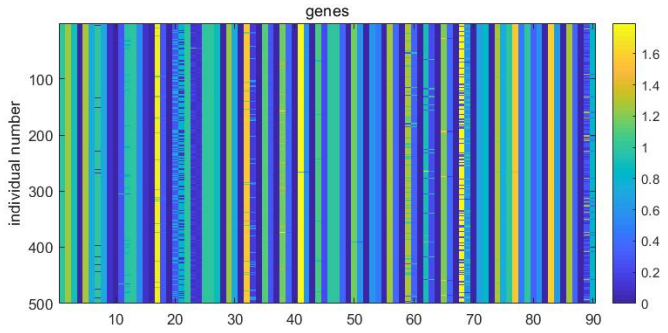
2) Uniform Change Mutation Operator

The simulation results after uniformly changing the mutation operator probability are shown in Fig. 12-13. It can be seen from Fig. 12 that when the uniform mutation operator probability is 0.01, the number of iterations continues to 6000. According to Fig. 12(c)-(d), the optimization effect of this simulation is obvious on the node with 3 loaded items, and the number of changes in the actual loaded times is relatively small. According to Fig. 12(a), when the number of iterations is 5120, a total of 8 items are loaded, and the area utilization rate is 50.76%.

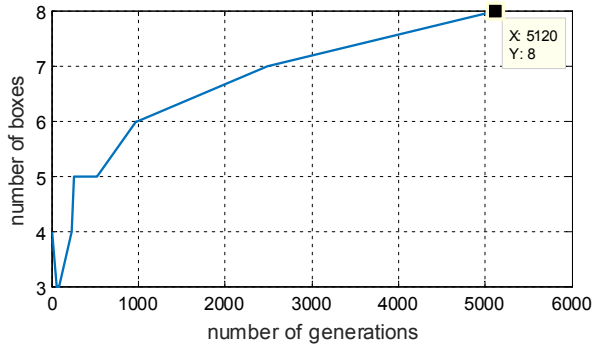
It can be seen from Fig. 13 that when the uniformly position mutation operator probability is 0.05, the number of iterations continues to 6000 times. According to Fig. 13(c)-(d), in the actual simulation process, the optimization is mainly concentrated on the nodes with 8, 9 and 10 loaded items, and the number of optimization iterations is mainly concentrated within 1000 times. According to Fig. 13(a), when the number of iterations is 5810, a total of 9 items are loaded, and the area utilization rate is 52.04%.



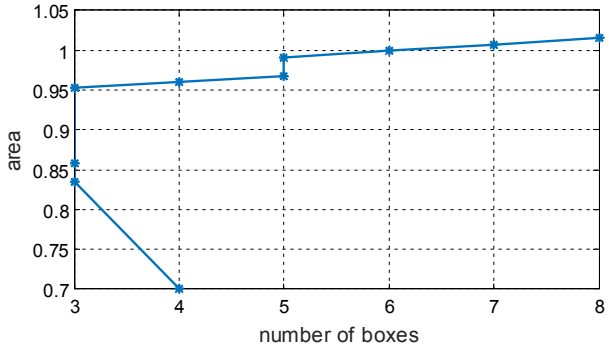
(a) Simulation diagram



(b) Gene diagram

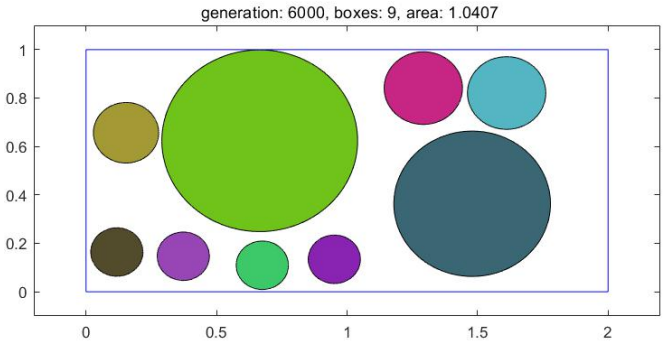


(c) Coordinate diagram of number of iterations and boxes

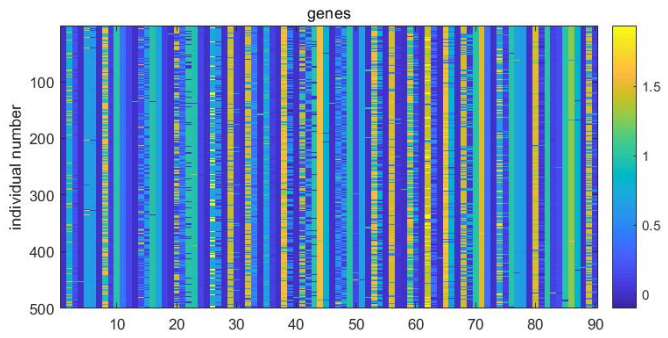


(d) Coordinate diagram of cumulative figure area and number of boxes

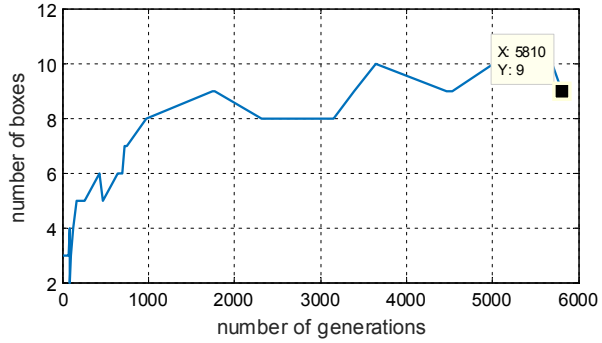
Fig. 12 Simulation results with uniform mutation operator of 0.01.



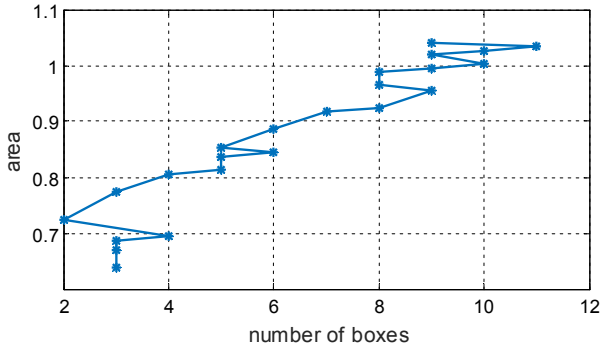
(a) Simulation diagram



（b）gene diagram



(c) Coordinate diagram of number of iterations and boxes



(d) Coordinate diagram of cumulative figure area and number of boxes

Fig. 13 Simulation results with uniform mutation operator of 0.05.

The performance comparison between single change location mutation probability and unified change mutation probability is shown in Table 2. Through the simulation, it is found that when the mutation operator probability is changed only to 0.05, the area utilization ratio is relatively small peak. However, it can be seen from the actual number of iterations that the number of iterations to generate a circle is more likely to be premature. When the mutation operator is uniformly changed, compared with the single change mutation operator probability, the occurrence of mutation is restricted in a certain way, and the space utilization rate is relatively reduced.

Two-dimensional bin-packing problem in rectangular and circular regions is solved based on genetic algorithm. It can be seen from the simulation results that when the mutation operator probability changes, the utilization rate of the box will also change randomly. Through 10 groups of simulation, it can be seen that the optimization process is mainly concentrated in the number of iterations within 1000 times. When the mutation operator increases, the utilization rate of area will also increase, but the optimal effect is that the mutation operator probability about 0.05 is the best. By comparing the results of circular packing and rectangular packing, it can be found that the utilization rate of circular packing case is lower than that of rectangular packing case under the same conditions. In the actual simulation process, the number of simulation iterations loaded with circles is easier to reach the convergence value, resulting in premature phenomenon.

TABLE 2 OPTIMIZATION PARAMETERS AND PERFORMANCE COMPARISON

Mutation operator	Position mutation operator	Visual mutation operator	Large Gaussian mutation operator	Small Gaussian mutation operator	Number of boxes	Utilization rate	Actual number of iterations	Total number of iterations
Single control	0.01	0.05	0.01	0.02	9	45.06%	4870	6000
	0.05	0.05	0.01	0.02	9	55.96%	4990	6000
	0.1	0.05	0.01	0.02	9	53.21%	5930	6000
Unified control	0.01	0.01	0.01	0.01	8	50.76%	5120	6000
	0.05	0.05	0.05	0.05	9	52.04%	5810	6000

V. CONCLUSIONS

The bin-packing problem is one of the classic combination optimization problem. The use of genetic algorithm to solve the two-dimensional bin-packing problem with rectangular and circular areas mainly relies on optimization by changing the genetic operators, etc. The simulation experiments are carried out by adopting 30 individuals, and the research is carried out by changing the mutation probability of the mutation operators on the basis of determining the fitness function. The main purpose of changing the mutation operator is to prevent the occurrence of premature phenomena. It can be seen from the simulation experiment results that the actual simulation effect is relatively good when the mutation operator is 0.05. The optimization of simulation process mainly focuses on the number of iterations within 1000 times.

REFERENCES

[1] T. G. Crainic, G. Perboli, W. Rei, and R. Tadei, "Efficient Lower Bounds and Heuristics for the Variable Cost and Size Bin Packing Problem," *Computers & Operations Research*, vol. 38, no. 11, pp. 1474-1482, 2011.

[2] A. Scholl, R. Klein, and Jürgens Christian, "BISON: a Fast Hybrid Procedure for Exactly Solving the One-dimensional Bin Packing Problem," *Computers & Operations Research*, vol. 24, no. 7, pp. 627-645, 1997.

[3] D. Pisinger, and M. Sigurd, "Using Decomposition Techniques and Constraint Programming for Solving the Two-dimensional Bin Packing Problem," *Informs Journal on Computing*, vol. 19, no. 1, pp. 36-51, 2007.

[4] J. N. D. Gupta, and J. C. Ho, "A New Heuristic Algorithm for the One-dimensional Bin-packing Problem," *Production Planning & Control*, vol. 10, no. 6, pp. 598-603, 1999.

[5] B. Xia, and Z. Tan, "Tighter Bounds of the First Fit Algorithm for the Bin-packing Problem," *Discrete Applied Mathematics*, vol. 158, no. 15, pp. 1668-1675, 2010.

[6] K. Fleszar, and C. Charalambous, "Average-weight-controlled Bin-oriented Heuristics for the One-dimensional Bin-packing Problem," *European Journal of Operational Research*, vol. 210, no. 2, pp. 176-184, 2011.

[7] J. B. G. Frenk, and G. Galambos, "Hybrid Next-fit Algorithm for the Two-dimensional Rectangle Bin-packing Problem," *Computing*, vol. 39, no. 3, pp. 201-217, 1987.

[8] M. Monaci, and P. Toth, "A Set-covering-based Heuristic Approach for Bin-Packing Problems," *INFORMS Journal on Computing*, vol. 18, no. 1, pp. 71-85, 2006.

[9] M. Haouari, and M. Serairi, "Heuristics for the Variable Sized Bin-packing Problem," *Computers & Operations Research*, vol. 36, no. 10, pp. 2877-2884, 2009.

[10] J. E. Hayek, A. Moukrim, and S. Negre, "New Resolution Algorithm and Pretreatments for the Two-dimensional Bin-packing Problem," *Computers & Operations Research*, vol. 35, no. 10, pp. 3184-3201, 2008.

[11] L. Babel, B. Chen, H. Kellerer, and V. Kotov, "Algorithms for On-line Bin-packing Problems with Cardinality Constraints," *Discrete Applied Mathematics*, vol. 143, no. 1-3, pp. 238-251, 2004.

[12] A. Singh, and A. K. Gupta, "Two Heuristics for the One-dimensional Bin-packing Problem," *Operations Research-Spektrum*, vol. 29, no. 4, pp. 765-781, 2007.

[13] T. Osogami, and H. Okano, "Local Search Algorithms for the Bin Packing Problem and Their Relationships to Various Construction Heuristics," *Journal of Heurs*, vol. 9, no. 1, pp. 29-49, 2003.

[14] J. E. Lewis, R. K. Ragade, A. Kumar, and W. E. Biles, "A Distributed Chromosome Genetic Algorithm for Bin-packing," *Robotics & Computer Integrated Manufacturing*, vol. 21, no. 4-5, pp. 486-495, 2005.

[15] M. J. Brusco, and G. M. T. W. Jacobs, "A Morph-based Simulated Annealing Heuristic for Modified Bin-packing Problem," *Journal of the Operational Research Society*, vol. 48, no. 4, pp. 433-439, 1997.

[16] T. G. Crainic, G. Perboli, and R. Tadei, "TS2PACK: A Two-level Tabu Search for the Three-dimensional Bin Packing Problem," *European Journal of Operational Research*, vol. 195, no. 3, pp. 744-760, 2009.

[17] C. G. Righini, "An Optimization Algorithm for the Ordered Open-End Bin-Packing Problem," *Operations Research*, vol. 56, no. 2, pp. 425-436 2008.

[18] M. Gabli, E. M. Jaara, and E. B. Mermri , "A Genetic Algorithm Approach for an Equitable Treatment of Objective Functions in Multi-objective Optimization Problems," *IAENG International Journal of Computer Science*, vol. 41, no. 2, pp. 102-111, 2014.

[19] S. D. Dao, K. Abhary, and R. Marian, "Maximising Performance of Genetic Algorithm Solver in Matlab," *Engineering Letters*, vol. 24, no. 1, pp. 75-83, 2016.

Nan Wang is a doctoral candidate of College of Economics and Management, Nanjing University of Aeronautics and Astronautics, Nanjing, 211106, P. R. China; Senior research fellow, Space Communication Center, Beijing, 100048, P. R. China. His main research interest is big data mining and optimization algorithm.

Jie-Sheng Wang received his B. Sc. and M. Sc. degrees in Control Science from University of Science and Technology Liaoning, China in 1999 and 2002, respectively, and his Ph. D. degree in Control Science from Dalian University of Technology, China in 2006. He is a professor and Doctorr's Supervisor of School of Electronic and Information Engineering, University of Science and Technology Liaoning. His main research interest is modeling of complex industry process, intelligent control and computer integrated manufacturing.

Yong-Xin Zhang is an undergraduate of School of Electronic and Information Engineering, University of Science and Technology Liaoning, Anshan, 114051, P. R. China. His main research interest is modeling and optimization of complex process.

Tian-Zhu Li is a professor of School of Business Administration, University of Science and Technology Liaoning, Anshan, 114051, P. R. China. He is a professor and Master's Supervisor in School of Business Administration, University of Science and Technology Liaoning. His main research interest is business administration and optimization.