2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

Optimization Algorithm for Rectangle Packing Problem Based on Varied-factor Genetic Algorithm and Lowest Front-Line Strategy

Haiming Liu, Jiong Zhou，Xinsheng Wu, Peng Yuan

***Abstract*—Rectangle packing problem exists widely in manufacturing processes of modern industry, such as cutting of wood, leather, metal and paper, etc. It is also known as a typical NP-Complete combinatorial optimization problem with geometric nature, which contains two sub-problems, parking problem and sequencing problem of rectangles. Considering the features of the problem, this paper proposes an optimization algorithm based on an improved genetic algorithm (GA), combined with a lowest front-line strategy for parking rectangles on the sheet. The genetic algorithm is introduced to determine packing sequence of rectangles. To avoid premature convergence or falling into local optima, the traditional GA is improved by changing genetic factors according to quality of solutions obtained during evolution. Numerical experiments were conducted to take an evaluation for the proposed algorithm, along with a comparison with another algorithm. The simulation results show that the proposed algorithm has better performance in optimization results and can improve utilization rate of material effectively.**

I. INTRODUCTION

RECTANGLE packing problem is the most studied issue in the field of two-dimension packing problem. It studies how to pack a set of rectangles with various sizes on a specified rectangle sheet without overlap. The optimization target of packing problem is to minimize the wasted material and maximize the utilization rate. Application of rectangle packing exists widely in manufacturing processes of cutting, blanking and machining of various materials in modern industry, such as wood, leather, metal and paper, etc. Thus, it is an important means for manufacturing enterprises to achieve material savings and cost reduction that using advanced theory and technology to realize packing optimization.

According to computational complexity theory, rectangle packing problem is a typical NP-Complete combinatorial optimization problem. For such problem, computation complexity will grow explosively with increase of scale of problem. Therefore, it is hard to obtain the optimal solution of the problem in a reasonable time. In the meanwhile, packing problem is also a layout problem with geometric nature, which makes it different from general combinatorial optimization problem. It is the key point that how to design and construct effective strategy and algorithm to find a satisfactory solution in an acceptable time.

There were many researches on rectangle packing problem in the past years and various algorithms were presented for problem solving. In these studies, most researchers decomposed the problem into two sub-problems as follows,

(1) Parking problem of rectangles, i.e. how to find out the most suitable position where the rectangles are place on the sheet. The sub-problem performs as a geometric layout problem.

(2) Sequencing problem of rectangles, i.e. how to determine packing sequence of each rectangle in packing process. This sub-problem performs as a combinatorial optimization problem.

Current researches on rectangle packing problem are focusing on problem-solving of the above sub-problems. Corresponding algorithms were designed to solve the sub-problems respectively. For parking problem of rectangles, a certain packing algorithms were put forward, such as BL algorithm[1], improved BL[2, 3], mate algorithm of surplus rectangles[4], skyline algorithm[5] and rectangle combination[6]. For sequencing problem of rectangles, intelligent optimization algorithms were considered in most literatures, such as heuristic algorithm[7], genetic algorithm[8, 9], simulated annealing algorithm[10], ant-colony algorithm[11] and particle swarm algorithm[12]. The solving methods of two sub-problems can be combined to solve the whole problem.

As an evolutionary algorithm, genetic algorithm has many advantages for solving sequencing problem of rectangles, such as its global search capability and clear structure. But, traditional GA also has some disadvantages, such as premature convergence. Also, to a large extent, performance of GA is influenced by design of evolution mechanism, such as selection, crossover and mutation of solutions. In this paper, we propose an optimization algorithm based on an improved GA, combined with a lowest front-line strategy for parking rectangles on the sheet. The genetic algorithm is introduced to determine packing sequence of rectangles. To avoid premature convergence or falling into local optima, idea of varied-factor is introduced in the traditional GA and reasonable evolution mechanism is design to improve search performance of algorithm.

LIU Haiming is with the College of Automation Science and Engineering, South China University of Technology, Guangzhou, 510640 PRC (phone: 080-8711-4489; e-mail: hmliu@ scut.edu.cn).

ZHOU Jiong is with the College of Automation Science and Engineering, South China University of Technology, Guangzhou, 510640 PRC (e-mail: 895531636@ qq.com).

WU Xinsheng is with the College of Automation Science and Engineering, South China University of Technology, Guangzhou, 510640 PRC (e-mail: auxswu@ scut.edu.cn).

YUAN Peng is with the College of Automation Science and Engineering, South China University of Technology, Guangzhou, 510640 PRC (e-mail: pangyuan@ scut.edu.cn).

This work was partly supported by National High Technology Research and Development Program of China (No. 2012AA041312) and the Fundamental Research Funds for the Central Universities of SCUT (No. 2014ZZ0033).

II. Problem Description

According to different technological requirement in practice, rectangle packing problem has different descriptions. The generalized description of the problem can be stated as follows: For a set of given rectangles with different sizes, to find the best packing plan under specific requirements to place all the rectangles to rectangular sheet with fixed width but unlimited height(or fixed height but unlimited width), and maximize utilization of the sheet. Fig. 1 shows a packing result of eight rectangles on a sheet with fixed width (the shaded parts represent the wasted material).

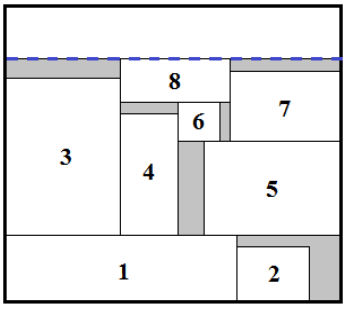

Fig. 1 Schematic diagram of rectangle packing

Generally, packing of rectangles should satisfy the following constraints:

(1) There is no overlap between two arbitrary rectangles.

(2) Every rectangle could not exceed the borders of the sheet.

(3) Rotation of rectangles is allowable, but the rotation angle can only be 90 degrees, which guarantees the borders of angle parallel to the borders of sheet.

Given a sheet with a width of W, x_i and y_i denote the bottom-left coordinates of rectangle $p_i(1\le i\le n)$ in X direction (horizontally) and Y direction (vertically) respectively after it was placed on the sheet, l_i and h_i denote the sizes of rectangle $p_i(1\le i\le n)$ in X direction and Y direction respectively when being placed on the sheet, S denotes sum of areas of all the rectangles, and H denotes the Y-coordinate of the highest horizontal line of outer contour formed by all rectangles after a whole packing process. The utilization of sheet is defined as ratio of sum of areas of all rectangles to area of used sheet and is denoted as U. Under the above definition, the mathematical model of rectangle packing problem can be formulated as follows:

$$\begin{aligned} &Max\ \ U=\frac{\sum_{i=1}^{n} l_i h_i}{WH}, \\ &s.t.\begin{cases} 0\le x_i\le W \\ 0\le x_i+l_i\le W \\ y_i\ge 0 \end{cases}, \ 1\le i\le n \end{aligned} \tag{1}$$

Thus, the optimization objective of rectangle packing problem is to search for the best packing scheme to maximize the utilization ratio of sheet, i.e. U. Obviously, for a set of given rectangles and a rectangular sheet with specific size, the packing scheme depends on two factors: one is the way a rectangle parks around those rectangles which were already placed, and the other is the packing sequence of each rectangles. The optimization proposed later in this paper will present the solving method for the two problems.

III. Description of Optimization Algorithm

Considering geometric feature of packing problem, we apply a parking algorithm based on lowest front-line strategy to place every rectangle; for its combinatorial optimization feature, we apply an improved genetic algorithm to solving the sequencing problem. A combination algorithm of the two algorithms is constructed to solve the rectangle packing problem.

A. Parking algorithm based on lowest front-line strategy

In the past literatures, BL algorithm or improved BL algorithm were often used to park rectangles for its simplicity and feasibility. But numerical tests showed that they had common disadvantages that "peaks" of rectangles often occur while packing rectangles, which made optimization performance deteriorate in some cases. To avoid the "peak" problem, we apply a parking algorithm based on lowest front-line strategy to place every rectangle. The concept "front-line" means the outer contour formed by borders of the sheet and the rectangles already placed on the sheet, which could be expressed by a set of horizontal lines. The lowest front-line strategy always gives priority to the lowest lines on the front-line to park the next rectangle to be placed. The steps of the parking algorithm are:

Step 1: Initialize the front-line. At the beginning of packing, the front-line is just the bottom border of the sheet (i.e. the initial front-line contains only one horizontal line).

Step 2: Place one rectangle on the sheet based on lowest front-line strategy. When placing the ith rectangle, named p_i, we first select a horizontal line with lowest height from the front-line, then judge whether the width of the chosen line are equal or bigger than the width of p_i. If the condition is true, we place the rectangle p_i on that line and park it to the left side. If the condition is false, we raise the height of the lowest line to that of the second lowest horizontal line in the front-line and repeat the judgement stated previously until a line from the front-line wide enough to place the rectangle.

In this step, some special cases should be noticed. First, if there are more than one horizontal line with lowest height, we choose the line to conduct judgement from low to high according to their X coordinates in the sheet. When a line satisfies, the placement of the rectangle is conducted and the selection is terminated. Second, if the raised horizontal line happens to close to the second lowest line, these adjacent horizontal lines with same height should be merged into one horizontal line to provide a wider space.

Step 3: Update the front-line. With a new rectangle placed on the sheet, the contour of placed rectangles inevitably changes. So, the horizontal lines on the front-line should be updated. Some old line may be cut into shorter lines, and some new line may be added in. Also, like the case mention in step 2, some adjacent lines with same height should be merged into one line. In this step, we are to obtain a new set of horizontal line of the new front-line.

Step 4: If all rectangles were placed, then terminate the

algorithm; otherwise, choose the next rectangle to be placed, go back to step 2 and continue.

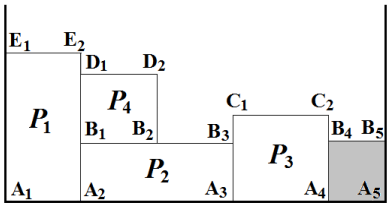

Fig. 2 Parking algorithm based on lowest front-line strategy

Fig. 2 shows a demonstration of parking algorithm based on lowest front-line strategy. Initially, the set of horizontal line on the front-line is $\{A_1A_5\}$. Obviously, A_1A_5 is chosen to place rectangle P_1, and the new front-line is $\{E_1E_2, A_2A_5\}$ after placement of P_1. Similarly, the lowest line A_2A_5 is then chosen to place rectangle P_2, and the resulted front-line is $\{E_1E_2, B_1B_3, A_3A_5\}$. In same mode, rectangle P_3 is placed on the line A_3A_5 and $\{E_1E_2, B_1B_3, C_1C_2, A_4A_5\}$ is obtained. At this point, the lowest line in the front-line is A_4A_5, while its width is not enough to place rectangle P_4. Thus, A_4A_5 is raised to be aligned with B_1B_3, the second lowest line in front-line. Because there are two lowest line (A_4A_5 and B_1B_3), search is conducted and line B1B3 is found out to place P_4. The final front-line is then $\{E_1E_2, D_1D_2, B_2B_3, C_1C_2, B_4B_5\}$ after placement of P_4.

B. Improved Genetic Algorithm based on varied factors

Traditional genetic has disadvantage of premature convergence. One of the causes is the roulette method which is often used to generate child solutions from parents. For example, some chromosomes with good fitness reproduce rapidly, which cause premature convergence; randomness of roulette may miss some good chromosomes and cause population degradation. Rectangle packing problem is a kind of discrete optimization problem. In the discrete solution space, the optimal solution may not always exist in the neighborhood of a sub-optimal solution, while it may close to a worse solution. For these reasons, we improve the genetic algorithm from two sides: first, we impose different evolution method on solutions with different fitness, which guarantees diversity of population; second, we apply different set of genetic factors based on evolution performance of solutions, which introduces self-adjusting capability for the algorithm and make the algorithm approach much closer to the optimal solution in a limited number of iterations.

1) Chromosome encoding and fitness function

In the genetic algorithm, chromosome encoding is to transfer possible solutions in original solution space to solutions which can be handled by GA. In GA, a chromosome is used to represent a possible solution, which can be encoded in binary, real or symbol. Here we use decimal encoding method: first we number each rectangle with a unique decimal number continuously, and then we construct a numerical sequence with numbers of all rectangles, in which the order of appearance of a numbered rectangle represents its packing sequence. If the number of every rectangle is regarded as a gene, the numerical sequence is just a chromosome (i.e. a solution in solution space). The coding way could be described mathematically: for n rectangles $\{p_1, p_2,..., p_n\}$, where $p_i(1\le i\le n)$ is a decimal number corresponding to a specific rectangle, the set of possible chromosomes is $C=\{(p_{k1}, ... p_{ki},..., p_{kj},..., p_{kn})| 1\le ki, kj\le n , ki \ne kj \}$.

According to equation (1), we choose the fitness function as $f(P)=S/S_P$, where P denotes a packing scheme, S is sum of all rectangles and S_P is area of used sheet, part under the highest horizontal line on the front-line. obviously， $0< f(P)\le 1$.

2) Operation of selection, crossover and mutation

Selection, crossover and mutation are important method of solution evolution in GA. To guarantee population diversity, different from traditional GA, we impose different operations on solutions with different fitness. First, we sort the solutions in parent population by fitness of them in descending order. Then, solutions with better fitness (i.e. solutions in the front of the sorted sequence) are reserved to be a part of solutions in children population. Finally, the rest solutions, which have worse fitness, are used to carry out crossover or mutation to generate other children solutions.

Provided that the size of population in GA is N and remains constant, the operations of selection, crossover and mutation are described as follows.

(1) Selection

In our GA, selection is used to retain better solutions to children population. The method of selection is to retain the solutions of top p_sN fitness in the parent population to the next population, where p_s ($0<p_s<1$) is a preset selection factor and N is size of population. The roulette method is abandoned for avoiding possible loss of good solutions.

(2) Crossover

Crossover is the most direct and effective method to generate new solutions, which plays a central role in GA. In our GA, we choose PMX crossover method proposed by Goldberg and Lingle[13] to generate new solutions. The main point of PMX crossover is choosing randomly two solutions in the rest solutions which are not chosen in selection stage to generate new solutions by two-point crossover method. For two solutions, e.g. $S1$ and $S2$, the rules of generating new solutions by PMX crossover are described as follows:

Step 1: generate two crossover points randomly in $S1$ and $S2$, and define the matching parts, as shown in Fig. 3.

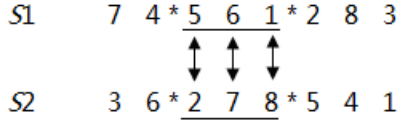

Fig. 3 Generate matching parts

In Fig. 3, the asterisk(*) is crossover points generated randomly, and parts between crossover points are matching parts of two solutions, which are labeled with underlines. There is a one-to-one correspondence between the genes (i.e. numbers) in the matching parts of two solutions in accordance with their locations. For example, the number 5, 6 and 1 in $S1$ are corresponding to 2, 7 and 8 in $S2$ respectively.

Step 2: Exchange the two matching parts and obtain two "new solutions" $S1$' and$S2$', as shown in Fig. 4. The "new solutions" may be illegal solution for they may have same genes in their own sequence, for example, the genes labeled with boxes in Fig. 4.

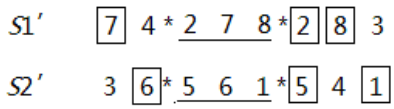

Fig. 4 "new solutions" obtained by exchanging matching parts

Step 3: for "new solutions" obtained in step 2, replace relevant genes according to their correspondence and obtain two legal new solutions *S*1'' and *S*2''. For "new solutions" in Fig. 4, the illegal genes in *S*1', 7, 2 and 8, are replaced respectively by corresponding genes, i.e. 6, 5 and 1, which is designated in Fig. 3. Then, a new legal solution, *S*1'', is obtained. The other new solution, *S*1'', could be also obtained in same way. The result of replacement of genes is shown in Fig. 5.

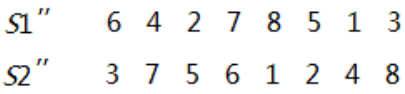

Fig. 5 Legal new solutions after gene replacement

By the crossover operation mentioned above, every set of two solutions selected randomly from parent population can generate two new solutions. The quantity of new solutions generated is limited to p_cN, where p_c ($0<p_c<1$) is the preset selection factor and N is size of population.

(3) Mutation

Mutation is another way to generate new solutions and is a effective supplement of crossover. Moreover, it could keep diversification of solutions to escape local optima, which will improve global search performance of GA. Considering features of packing problem, we design two mutations to generate new solutions: one is exchange mutation and the other is rotation mutation.

Exchange mutation means exchanging locations of two genes selected randomly in one parent solution, by which a new solution is generated.

Rotation mutation is used to determine whether a rectangle is rotated by 90 degrees when being placing on the sheet. To conduct rotation mutation, we first select a parent solution randomly, then generate a random position between [1, n], where n is the length of a chromosome. The gene (corresponding to a specific rectangle) on the position should change its original rotation direction, i.e., 0 changed to 90 or 90 changed to 0. In order to contain rotation direction in a chromosome, we extend encoding method mentioned previously by using sign of plus or minus to specify whether a rectangle should be rotated. In the simulation test, we use plus for no rotation (0 degree) and minus for rotation by 90 degrees. So, we can just apply sign reversing to a gene selected randomly to conduct rotation mutation.

Both of the mutations could generate a new solution from a parent solution. We define the quantities of new solutions generated by exchange mutation and rotation mutation are $p_{me}N$ and $p_{mr}N$ respectively, where p_{me} ($0<p_{me}<1$) and p_{mr} ($0<p_{mr}<1$) denotes a preset exchange mutation factor and a preset rotation mutation factor respectively.

It is noted that, in order to maintain a constant population size of each generation, the factors of selection, crossover and mutation should satisfy the condition of $p_s+p_c+p_{me}+p_{mr}=1$.

3) GA based on varied-factor

Researches showed that convergence rate and solution quality of GA depends on selection of genetic factors to a great extent. Generally, selection factor has a main impact on convergence rate of GA, while crossover factor and mutation factor play important part in searching for better solutions in the solution space. From this point, we consider apply different factor settings at different stages of algorithm, by which the algorithm could be self-adjusting and have better performance than fixed factor setting. The idea of varied-factor GA is described as follows:

At the initial stage of algorithm, selection factor p_s is set to a bigger value（$0.6\le p_s<1$），the other three factors, p_c, p_{mc} and p_{mr}, are set to smaller values。A bigger selection factor will make algorithm converge in a short time and approach to a sub-optimal solution. When the algorithm is searching repeatedly in a subspace and the best solution found has not been improved in consecutive generations, it means the algorithm has fallen into local optimal and can not extend the search to much wider space. In this case, it needs to adjust the genetic factors to prevent the algorithm from premature convergence. Adjustment of genetic factors is somewhat opposite to the initial stage, that is to set p_s to a smaller value($0<p_s\le 0.4$), and p_c, p_{mc} and p_{mr} are set to bigger values. The purpose of that is to let more parent solutions join the operations of crossover and mutation, which could not only generate more new solutions but also extend search space.

It is noted that the stages of algorithm vary with changes of the best solution found and settings of the judgement conditions. In practical execution of algorithm, these two stages execute alternately, which prevents the algorithm from being subject to fixed factors.

C. *Optimization algorithm for rectangle packing problem*

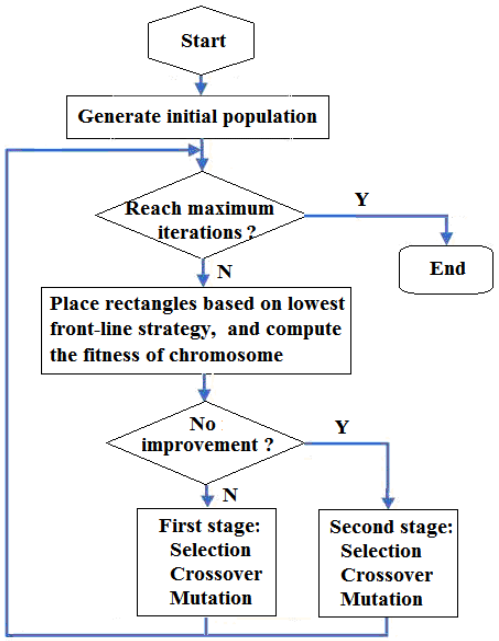

Fig. 6 Optimization algorithm for rectangle packing problem

Combined with the lowest front-line parking algorithm, the varied-factor GA is constructed to solve the rectangle packing problem. The flow chart of the combined algorithm is shown in Fig. 6. In the flow chart, every chromosome generated in each generation of population by varied-factor GA will be interpreted as a packing sequence of rectangles. Then, the parking algorithm based on lowest front-line will be applied to these packing sequences to place rectangles one by one. Fitness of each chromosome will be computed and evaluated by the cost function mentioned previously for each layout result.

IV. Simulation Results

To evaluate the performance of optimization algorithm proposed in this paper, we programmed under Visual C++ 2008 to implement this algorithm.

First, we made a comparison with the algorithm proposed in literature [8], which applied traditional GA to solve the problem, under the simulation example. In the example, 20 kinds of rectangles with different sizes are given (the quantity of rectangles is 59), and the width of the sheet is 400. Details of rectangles of the example are shown in Table I.

In our simulation test, the parameters involved in our algorithm are listed in Table II. The condition of switching to the other search stage (corresponding to different set of genetic factors) is whether the best solution found has not been improved in 5 generations. The condition of termination is that the times of iterations reach the preset maximum iteration.

The optimization results of the two algorithms are shown in Table III. In the table, *Average* means the average utilization rate of 50 times of computation results. *Best* means the best results that the algorithms obtained in 50 computations. *AverageTime* means average time for computation.

For further evaluation of the algorithms, we took simulation tests on 10 groups of examples generated randomly under the same parameters. The test data are shown in Table IV.

The comparison results shown in Table IV demonstrate that the algorithm proposed in this paper obtains higher average and best utilization rates of sheet than algorithm in [8] for almost every tested example. The simulation test indicates that varied genetic factors in different stages of genetic algorithm would help improve local or global search ability and have more chance to reach optimal solution during the optimization process. Furthermore, under the maximum iteration of 300 with 100 solutions in each population, the average computation times needed are not so long for all the examples, which means the algorithm used here would have satisfactory response time in practice.

TABLE I
Details of Retangles of Simulation Example in [8]

No. of Rectangle types	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Length	25	18	79	121	29	64	36	48	11	46	55	87	39	31	41	78	19	63	10	50
Height	36	24	84	30	48	98	21	59	17	121	22	41	72	25	65	24	11	36	30	61
Quantity	4	5	3	4	11	2	2	3	2	2	1	2	2	2	2	3	2	2	3	2

TABLE II
Parameters Involved in Algorithm

Size of popul-ation	Maxi-mum itera-tion	First Stage				Second Stage			
		p_s	p_c	p_{me}	p_{mr}	p_s	p_c	p_{me}	p_{mr}
100	300	0.8	0.1	0.05	0.05	0.2	0.4	0.2	0.2

TABLE III
Optimization Results(I)

Algorithm in [8]		Algorithm in this paper		
Average	*Best*	*Average*	*Best*	*AverageTime*(s)
84.59%	85.92%	85.51%	87.75%	26.3

V. Conclusion

This paper presents an effective optimization algorithm based on lowest front-line strategy and varied-factor GA for solving rectangle packing problem. The method of varying genetic factors in different stages of algorithm can improve search performance in the solution space and prevent premature convergence. The evaluation data show that the proposed algorithm can obtain better results than traditional

TABLE IV
Optimization Results(II)

Group No.	Rectangle		Algorithm in [8]		Algorithm in this paper		
	Kinds	Quantity	*Average*	*Best*	*Average*	*Best*	*AverageTime*(s)
1	13	80	84.97%	86.04%	85.36%	86.99%	37.5
2	14	78	85.54%	86.46%	85.59%	87.57%	36.3
3	15	79	84.33%	85.98%	84.94%	85.76%	38.6
4	15	98	85.11%	85.46%	85.54%	87.71%	45.9
5	16	76	83.75%	85.47%	84.94%	87.11%	35.7
6	16	92	83.79%	84.78%	84.44%	85.72%	42.1
7	18	113	82.05%	82.33%	83.08%	84.12%	53.4
8	20	112	83.52%	84.72%	84.19%	84.96%	51.4
9	21	99	84.77%	85.75%	85.29%	86.70%	42.5
10	29	142	85.49%	86.34%	85.82%	88.01%	62.7

GA. It should be mentioned that combinations of genetic factors used in our algorithm are relatively fixed, which makes the algorithm has different performance when dealing with different examples. In the future work, we will make in-depth analysis to the influence of different combinations of genetic factors on performance of the algorithm and seek a way to making the algorithm have the ability of self-adaptive.

REFERENCES

[1] Jakobs S. "On the genetic algorithms for the packing of polygons," *European Journal of Operational Research*, 1996, 88: 165- 181.

[2] Leo Ho Wai Yeung, K S Wallace Tange. "A hybrid genetic approach for garment cutting in the clothing industry," *IEEE Transactions on Industrial Electronics*, 2003, 50(3): 449- 455.

[3] Tang Kwok-wah, Tang Wallace Kit-sang. "Metal cutting with hybrid genetic algorithm," *3rd IEEE International Conference Industrial Informatics*, 2005: 735- 739.

[4] LIANG Lidong, YE Jiawei, WEI dong. "Mate Algorithm of Surplus Rectangle on Layout of the Parts of Ships," *Ship & Ocean Engineering*, 2008, vol. 37, no. 4, pp. 7-9.

[5] Lijun Wei, Wee-Chong Oon, Wenbin Zhu, Andrew Lim. "A skyline heuristic for the 2D rectangular packing and strip packing problems," *European Journal of Operational Research*, 2011, 215: 337-346.

[6] Christoforos Charalambous,Krzysztof Fleszar. "A constructive bin_oriented heuristic for the two-dimensionalbin packing problem with guillotine cuts," *Computers & Operations Research,*. 2011, 38: 1443-1451.

[7] Hopper E, Turton B C H. "An empirical investigation of meta-heuristics and heuristics algorithms for a 2D packing problem," *European Journal of Operational Research*, 2001, 128: 34- 57.

[8] Gong Zhihui, Huang Xingmei. "Study on Improvement of Optimization Algorithm for Two-dimensional Rectangle Pakcing," *Journal of Hunan University(Natural Sciences)*, 2003, 30(3): 47-49.

[9] Chaouiya C. "Petri net modelling of biological networks," *Brief Bioinform*, 2007, 30(11): 1889-1900.

[10] Dereli T, Das G S. "A hybrid simulated-annealing algorithm for two-dimensional strip packing problem," *Adaptive and Natural Computing Algorithms*, 2007: 508-516.

[11] Salto C, Leguizamón G, Alba E, et al. "Evolutionary and Ant Colony Optimization Based Approaches for a Two-Dimensional Strip Packing Problem," *Natural Intelligence for Scheduling, Planning and Packing Problems*, 2009: 245-266.

[12] WANG Shan-shan. "Application of Chaotic Discrete Particle Swarm Algorithm in Rectangle Packing," *Computer Engineering*, 2010, 36(23): 174-176.

[13] Goldberg DE, Lingle R. "Alleles, loci, and the traveling salesman problem," *In: Grefenstette JJ (ed) Proceedings of an International Conference on Genetic Algorithms and Their Applications*. 1985: 154-159.