An Improved Heuristic Recursive Strategy Based on Genetic Algorithm for the Strip Rectangular Packing Problem

ZHANG De-Fu[1] CHEN Sheng-Da[1] LIU Yan-Juan[1]

Abstract An improved heuristic recursive strategy combining with genetic algorithm is presented in this paper. Firstly, this method searches some rectangles, which have the same length or width, to form some layers without waste space, then it uses the heuristic recursive strategies to calculate the height of the remaining packing order and uses the evolutionary capability of genetic algorithm to reduce the height. The computational results on several classes of benchmark problems have shown that the presented algorithm can compete with known evolutionary heuristics. It performs better especially for large test problems.

Key words Strip packing problems, heuristic, recursive, genetic algorithm

1 Introduction

Many industrial applications, which belong to cutting and packing problems, have been found. Each application incorporates different constraints and objectives. For example, in wood or glass industries, rectangular components have to be cut from large sheets of material. In warehousing contexts, goods have to be placed on shelves. In newspapers paging, articles and advertisements have to be arranged in pages. In the shipping industry, a batch of objects of various sizes has to be shipped to the maximum extent in a larger container, and a bunch of optical fibers has to be accommodated in a pipe with perimeter as small as possible. In very-large scale integration (VLSI) floor planning, VLSI has to be laid out. These applications have a similar logical structure, which can be modeled by a set of pieces that must be arranged on a predefined stock sheet so that the pieces do not overlap with one another, so they can be formalized as the packing problem [1]. For more extensive and detailed descriptions of packing problems, the reader can refer to [1∼3].

A two-dimensional strip rectangular packing problem is considered in this paper. It has the following characteristics: a set of rectangular pieces and a larger rectangle with a fixed width and infinite length, designated as the container. The objective is to find a feasible layout of all the pieces in the container that minimizes the required container length and, where necessary, takes additional constraints into account. This problem belongs to a subset of classical cutting and packing problems and has been shown to be non-deterministic polynomial (NP) hard[4, 5]. Optimal algorithms for orthogonal two-dimension cutting were proposed in [6, 7]. Gilmore and Gomory[8] solved problem instances to optimality by linear programming techniques in 1961. Christofides and Whitlock[9] solved the two-dimensional guillotine stock cutting problem to optimality by a tree-search method in 1977. Cung *et al.*[10] developed a new version of the algorithm proposed in Hifi and Zissimopolous that used a best-first branch-and-bound approach to solve exactly some variants of two-dimensional stock-cutting problems in 2000. However, these algorithms might not be practical for large problems. In order to solve large problems, some heuristic algorithms were developed. The most documented heuristics are the bottom-left (BL), bottom-left-fill (BLF) methods, and other heuristics[11∼13]. Although their computational speed is very fast, the solution quality is not desirable. Recently, genetic algorithms and its improved algorithms for the orthogonal packing problem were proposed because of their powerful optimization capability[14∼17]. Kroger[14] used genetic algorithm for the guillotine variant of bin packing in 1995. Jakobs[15] used a genetic algorithm for the packing of polygons using rectangular enclosures and a Bottom-left heuristic in 1996. Liu *et al.*[16] further improved it. Hopper and Turton[18] evaluated the use of the BLF heuristic with genetic algorithms on the nonguillotine rectangle-nesting problem in 1999. In addition, an empirical investigation of meta-heuristic and heuristic algorithms of the strip rectangular packing problems was given by [19]. Recently, some new models and algorithms were developed by [20∼26]. For example, quasi-human heuristic[20], constructive approach[21, 22], a new placement heuristic[23], heuristic recursion (HR) algorithm[24], and hybrid heuristic algorithms[25, 26] were developed. These heuristics are fast and effective, especially, the best fit in [23] not only is very fast, but also finds better solutions than some well-known metaheuristics.

In this paper, an improved heuristic recursive algorithm that combines with genetic algorithm is presented to solve the orthogonal strip rectangular packing problem. The computational results on a class of benchmark problems show that this algorithm can compete with known evolutionary heuristics, especially in large test problems.

The rest of this paper is organized as follows. In Section 2, a clear mathematical formulation for the strip rectangular packing problem is given. In Section 3, the heuristic recursive algorithm is presented, and an improved heuristic recursive algorithm (IHR) is developed in detail. In Section 4, the GA+IHR algorithm is proposed. Computational results are described in Section 5. Conclusions are summarized in Section 6.

2 Mathematical formulation of the problem

Given a rectangular board of given width and a set of rectangles with arbitrary sizes, the strip packing problem of rectangles is to pack each rectangle on the board so that no two rectangles overlap and the used board height is minimized. This problem can also be stated as follows.

Received June 20, 2006; in revised form October 24, 2006
Supported by Academician Start-up Fund (X01109), 985 Information Technology Fund (0000-X07204) in Xiamen University
1. Department of Computer Science, Xiamen University, Xiamen 361005, P. R. China
DOI: 10.1360/aas-007-0911

Given a rectangular board with given width W, and n rectangles with length l_i and width w_i, $1 \leq i \leq n$, let (x_{li}, y_{li}) denote the top-left corner coordinates of rectangle i, and (x_{ri}, y_{ri}) the bottom-right corner coordinates of rectangle i. Other symbols are similar to [24]. For all $1 \leq i \leq n$, the coordinates of rectangles must satisfy the following conditions:

1) $(x_{ri} - x_{li} = l_i$ and $y_{li} - y_{ri} = w_i)$ or $(x_{ri} - x_{li} = w_i$ and $y_{li} - y_{ri} = l_i)$;

2) For all $1 \leq j \leq n$, $j \neq i$, rectangle i and j can not overlap, namely, $x_{ri} \leq x_{lj}$ or $x_{li} \geq x_{rj}$ or $y_{ri} \geq y_{lj}$ or $y_{li} \leq y_{rj}$;

3) $x_L \leq x_{li} \leq x_R, x_L \leq x_{ri} \leq x_R$ and $y_R \leq y_{li} \leq h$, $y_R \leq y_{ri} \leq h$.

The problem is to pack all the rectangles on the board such that the used board height h is minimized.

It is noted that for orthogonal rectangular packing problems the packing process has to ensure the edges of each rectangle are parallel to the x- and y- axes, respectively, namely, all rectangles can not be packed aslant. In addition, all rectangles except the rectangular board are allowed to rotate 90 degrees.

3 IHR algorithm

The HR algorithm for the strip rectangular packing problem was presented in [24]. It follows a divide-and-conquer approach: break the problem into several subproblems that are similar to the original problem but smaller in size, solve the subproblems recursively, and then combine these solutions to create a solution to the original problem. The HR algorithm is very simple and efficient, and can be stated as follows:

1) Pack a rectangle into the space to be packed, and divide the unpacked space into two subspaces.

2) Pack each subspace by packing it recursively. If the subspace size is small enough to only pack a rectangle, then just pack this rectangle into the subspace in a straightforward manner.

3) Combine the solutions to the subproblems for the solution of the rectangular packing problem.

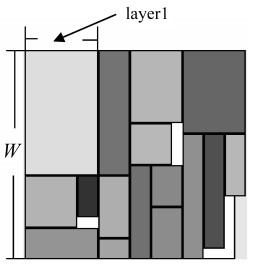
In order to enhance the performance of the HR algorithm, the author in [18] presented some heuristic strategies to select a rectangle to be packed, namely, the rectangle with the maximum area is given priority to pack. In detail, unpacked rectangles should be sorted by nonincreasing ordering of area size. The rectangle with maximum area should be selected to pack first if it can be packed into the unpacked subspace. In addition, the longer side of the rectangle to be packed should be packed along the bottom side of the subspace.

It is the disadvantage of the HR algorithm that may have waste space in each layer (See Fig. 1). In order to overcome this disadvantage, some layers without waste space are first considered. Some definitions are given to clearly describe the idea of the improved algorithm.

Definition 1. The reference rectangle is the rectangle that is packed firstly and can form one layer with other rectangles, and its short edge is the height of that layer.

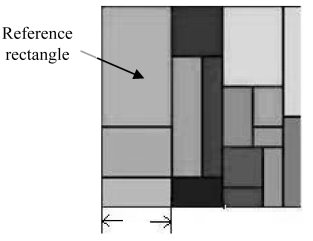
Definition 2. The combination layer is the layer that has no waste space, and the rectangles packed into it have the same height as the height of the layer and are spliced together one by one along the direction of W. The sum of the edge length of the rectangles along the W direction is the combination width.

From Fig. 1 to Fig. 4, each rectangle at the top of the container is the referee rectangle. The first layer in Fig. 1 is not a combination layer because it has waste space. The first layer in Fig. 2 is a combination layer because it has no waste space and the spliced rectangles have the same width or length as the height of the layer. Although the second layer in Fig. 2 has no waste space, it is not a combination layer because the two middle rectangles are not spliced along the direction of W. By the definition of the combination layer, the combination width is W. If some combination layers are found before further computation, then they may decrease the cost of computation because the rectangles already packed into these combination layers will not be considered in the future.



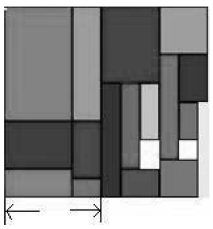
Number of combination layers = 0

Fig. 1



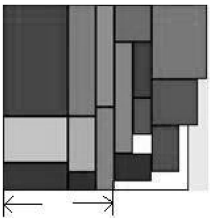
Number of combination layers = 1

Fig. 2



Number of combination layers = 2

Fig. 3



Number of combination layers = 3

Fig. 4

From the above discussion, it is very important to find the combination layer. Given a packing ordering, the procedure of finding the combination layer is given as follows.

Find the first unpacked and unreferenced rectangle as the reference rectangle, and put this rectangle into a two-dimensional array. Then seek downwards from the reference rectangle orderly. If one can find a rectangle whose length or width is equal to the width of the reference rectangle, then put this rectangle into the two-dimensional array, repeat this until a combination layer or no combination layer is found. Repeat the above process until all rectangles are packed otherwise no rectangle can be the reference rectangle. In this process, the number of rectangles, which have been packed in the current layers, must be recorded. Finally, the number of all the combination layers must be recorded. The steps of the combination operator can be stated as follows:

```
Combination ( )
  Repeat
    Find the first unpacked and unreferenced
    rectangle as the reference rectangle, and put this
    rectangle into a two-dimensional array;
    For i = current position to n
      If (the width or the length of rectangle i is
      equal to the width of the reference rectangle)
        If (combination width < W)
          Put the rectangle into the two-dimensional
          array;
        Else if (combination width = W)
          Pack all the rectangles of the
```

```
        two-dimensional array on the container;
        Break;
    Record the number of all packed rectangles;
  Until all rectangles are packed or no rectangle is the
  reference rectangle;
  Record the number (Num) of all the combination layers;
```

So, the IHR algorithm can be stated as follows:

Step 1. The combination layers are searched.

Step 2. Pack the remaining rectangles to the container by HR algorithm.

By intuition, the more the number of the combination layers is, the faster the computational speed is. However, it is not always true that more combination layers can obtain a better solution. From Fig. 1 to Fig. 4, we know that the best combination layer number is 1. Fig. 1 has no combination layer, but layer 1, layer 3, and layer 4 waste a little space. Fig. 2 has one combination layer and is the optimal solution. Fig. 3 has two combination layers but layer 3 and layer 4 waste a little space. Fig. 4 has three combination layers, but layer 4 and layer 5 waste a little space.

4 GA+IHR

4.1 Genetic algorithm (GA)

GA is a heuristic method used to find approximate solutions to hard optimization problems through application of the principles of evolutionary biology to computer science. It is modeled loosely on the principles of the evolution via natural selection, which use a population of individuals that undergo selection in the presence of variation inducing operators such as recombination (crossover) and mutation. In order to run GA, we must be able to create an initial population of feasible solutions. The initial population is very important for achieving a good solution. There are many ways to do this based on the form of the problems. The evolution starts from a population of completely random individuals and happens in generations. In each generation, the fitness of the whole population is evaluated. Multiple individuals are stochastically selected from the current population and are modified to form a new population, which becomes current population in the next iteration of the algorithm. Further detailed theoretical and practical descriptions of genetic algorithm, the interested reader can refer to [27].

Combining GA with IHR, the GA+IHR algorithm to solve the strip rectangular packing problem can be stated as follows:

```
GA+IHR( )
  Sort all rectangles by non-increasing ordering of area size;
  Combination ( );
  For i = 0 to Num
    Initialization ( );
      For j = 1 to Number
        For k = 1 to N/2
            Select two individuals in the parents
            randomly, then crossover with probability
            Pc or copy with probability (1 − Pc) to
            create two middle offspring;
            Mutate the middle offspring with
            probability Pm;
            Compare the parent and the middle offspring,
            if the fitness of the middle offspring is less than
            the parent's, we accept it as new offspring,
            otherwise we accept it with probability Pb or
            accept the parent with probability (1 − Pb);
        Select the best solution from the parents;
        Select the worse solution from the new
        generation;
        Replace the worse solution with the best
        solution;
      Save the best solution acquired from combination
      layer i to array A;
    Select the best solution from array A;
```

where Num is the number of all the combination layers; $Number$ is the iteration number of genetic algorithm; N is the number of population; $Pc = 0.8$, $Pm = 0.2$, but Pm will increase as the parent chromosomes become more alike, $Pb = 0.33$. The fitness value of genetic algorithm is calculated by HR algorithm. The required container length is the sum of combination height and current fitness value.

4.2 Initialization

The GA is used to optimize the solution for unpacked rectangles. The fitness value of GA is calculated by HR algorithm. In this paper, a string of integers, which forms an index into the set of rectangles, is used, and then the HR strategy is used to create the sequence of the first individuals, and the sequence is divided into two equal parts, the two parts of the first individuals are then permuted to obtain $N-1$ individuals (N is the size of population). The method of the permutation is to produce a point in each range randomly and exchange the position of the point for its neighbor. This initialization method can keep the diversification of each individual in the population. At the same time, it can keep the individual, which has better fitness. From the experiment results, we know that the method has better effect.

4.3 Crossover

The role of the crossover operator is to allow the advantageous traits to spread throughout the population such that the population as a whole may benefit from this chance discovery. The steps of the crossover operator are as follows:

1) Choose two individuals Parent 1 and Parent 2 from the parents randomly.

2) Get the items of individual Child 1 from Parent 1 and Parent 2 alternately. Namely, if the sequence number of the Child 1 is odd, find an item orderly in Parent 1 until the item is different from all the items in Child 1, otherwise find an item orderly in Parent 2 until the item is different from all the items in Child 1. When the number of Child 1 is n, a new individual is created successfully.

3) Similarly, get the items of individual Child 2 from Parent 2 and Parent 1 alternately. Namely, if the sequence number of the Child 2 is odd, find an item orderly in Parent 2 until the item is different from all the items in Child 2, otherwise find an item orderly in Parent 1 until the item is different from all the items in Child 2. When the number of Child 2 is n, a new individual is created successfully.

As an example of crossover, suppose two individuals are already selected:

Parent 1:	5	3	2	6	7	8	4	1
Parent 2:	8	6	5	1	7	3	2	4

According to the steps of the crossover operator, the sequences of the children can be obtained:

Child 1:	5	8	3	6	2	1	7	4
Child 2:	8	5	6	3	1	2	7	4

4.4 Mutation

In each individual A, two different points are chosen randomly, and the sequence within two points is inversed, then

in the appointed iteration step, judge the fitness of the new individual B, if the fitness of B is less than that of A, B is accepted.

As an example of mutation, suppose two mutation points (3 and 6) are already selected:

A: 2 4 5 8 7 1 3 6

After mutation, we can get the new individual

B: 2 4 1 7 8 5 3 6

Mutation is adaptive, that is, the mutation rate increases as the parent chromosomes become more alike.

4.5 Replacement

After the operations of crossover and mutation, a set of solutions are produced. For keeping the best fitness and quickening the speed of convergence, a best solution is selected from the set of solutions, and it is saved to the next generation in each iteration step.

5 Computational results

In order to compare the relative performance of the presented GA+IHR with other published heuristic and meta-heuristic algorithms, several test problems taken from the literature are used. Perhaps the most extensive instances given for this problem are found in [19], where 21 problem instances are presented in seven different sized categories ranging from 16 to 197 items. The optimal solutions of these 21 instances are all known. Table 1 (see next page) presents an overview of the test problem Class 1 from [19]. As we wanted to extensively test our algorithm, other test problems were generated at random. Table 2 shows an overview of the test problem Class 2 generated randomly with known optimal solution. The problem Class 2 can be accessed in [23].

In order to verify the performance of GA+IHR, two best meta-heuristic GA+BLF and SA+BLF[23], Best fit[13], and HR[19] are selected. The computational results are listed in Tables 3 and 4. 20 iterative times are chosen for GA and 80 iterative times are chosen in the mutation operation.

On this test problem Class 1, as listed in Table 3, Gap of GA+IHR ranges from 0.83 to 4.44 with the average Gap 2.06. The average Gaps of GA+BLF, SA+BLF, Best fit, and HR are 4.57, 4, 5.69, and 3.97, respectively. The average Gap of GA+IHR is lower than those of GA+BLF, SA+BLF, Best fit, and HR. And as listed in Table 4, the average running time of GA+IHR is also lower than those of GA+BLF and SA+BLF, but is larger than that of Best fit and HR. The packing results can be seen in Fig. 5 and Fig. 6, where L denotes the optimal height. The heights of C11, C12, C13, and C72 using GA+IHR are 20, 21, 21, and 241, respectively. GA+IHR can find the optimal heights for C11, C23, and C32.

In order to extensively test the performance of our algorithm for randomly generated instances, especially for larger instances, 12 problem instances ranging from 10 to 500 were generated at random. The computational results are listed in Table 5. For such problem, 100 iterative times are chosen for the GA and 10 iterative times are chosen in the mutation operation. Although our algorithm can obtain a better solution, it needs much time, especially for large problems. So for N12, 40 iterative times are chosen for the GA and 10 iterative times are chosen in the mutation operation. From Table 5, we observe that the running time is acceptable. What is more, the Gap is better than others.

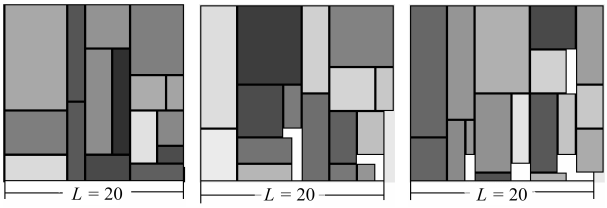


Fig. 5 Packed results of C1 for GA+IHR

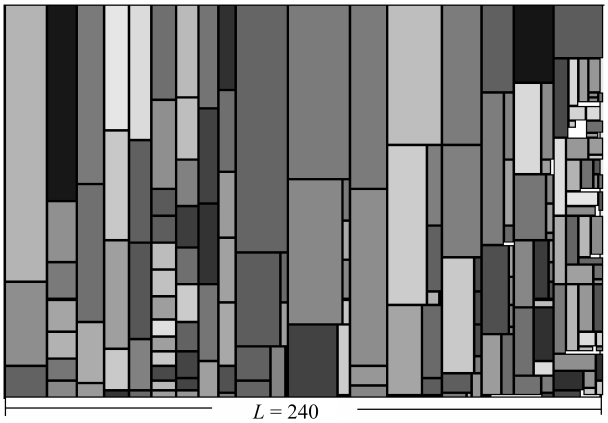


Fig. 6 Packed results of C72 for GA+IHR

6 Conclusions

In this paper, the GA+IHR algorithm for the orthogonal stock-cutting problem has been presented. IHR is very simple and intuitive, and can solve the orthogonal stock-cutting problem efficiently. GA is an adaptive heuristic search algorithm. It has the capability of global search within the solution space. The idea of combination layers to reduce the number of unpacked rectangles has been used. During the process of iteration search, HR is called repeatedly to calculate the height of an individual. As we know, finding the optimal solution is more difficult for the packing problem as increasing the size of problem. But it can be overcome by using the characteristic of GA. The computational results have shown that we can obtain the desirable solutions within acceptable computing time by combining GA with IHR. So GA+IHR can compete with other evolution heuristic algorithms, especially for large test problems, it performs better. So GA+IHR may be of considerable practical value to the rational layout of the rectangular objects in the engineering fields, such as the wood, glass, and paper industry, the ship building industry, and textile and leather industry. Future work is to further improve the performance of GA+IHR and minimize the influence of the parameters selection, and extend this algorithm for three-dimensional rectangular packing problems.

Table 1 Test problem Class 1

Problem category	Number of items: n	Optimal height	Object dimension
C1(C11,C12,C13)	16(C11,C13),17(C12)	20	20×20
C2(C21,C22,C23)	25(C21,C22,C23)	15	15×40
C3(C31,C32,C33)	28(C31,C33),29(C32)	30	30×60
C4(C41,C42,C43)	49(C41,C42,C43)	60	60×60
C5(C51,C52,C53)	73(C51,C52,C53)	90	90×60
C6(C61,C62,C63)	97(C61,C62,C63)	120	120×80
C7(C71,C72,C73)	196(C71,C73),197(C72)	240	240×160

Table 2 Test problem Class 2

Problem category	Number of items: n	Optimal height	Object dimension
N1	10	40	40×40
N2	20	50	30×50
N3	30	50	30×50
N4	40	80	80×80
N5	50	100	100×100
N6	60	100	50×100
N7	70	100	80×100
N8	80	80	100×80
N9	100	150	50×150
N10	200	150	70×150
N11	300	150	70×150
N12	500	300	100×300

Table 3 Gaps of GA+BLF, SA+BLF, Best fit, HR, and GA+IHR for the test problem Class 1

	C1	C2	C3	C4	C5	C6	C7	Average
GA+BLF	4	7	5	3	4	4	5	4.57
SA+BLF	4	6	5	3	3	3	4	4
Best fit	11.67	6.7	9.9	3.87	2.93	2.5	2.23	5.69
HR	8.33	4.45	6.67	2.22	1.85	2.5	1.8	3.97
GA+IHR	3.33	4.44	2.22	1.67	1.11	0.83	0.83	2.06

Table 4 Average running time of GA+BLF, SA+BLF, and GA+HR

	C1	C2	C3	C4	C5	C6	C7	Average
GA+BLF	4.61	9.22	13.83	59.93	165.96	396.46	3581.97	604.57
SA+BLF	3.227	11.064	18.44	52.13	530.15	1761.02	19274.41	3107.2
Best fit	0.0	0.0	0.0	0.00	0.003	0.005	0.007	0.005
HR	0	0	0.03	0.14	0.69	2.21	36.07	5.59
GA+IHR	0.88	1.52	2.24	9.56	30.04	65.56	426.04	76.55

Table 5 Gaps of GA+BLF, SA+BLF, Best fit, HR, and GA+IHR for the test problem Class 2

	n	Optimal height	GA+BLF		SA+BLF		BF Heuristic		GA+IHR	
			h	Time(s)	h	Time(s)	h	Time(s)	h	Time(s)
N1	10	40	40	1.02	40	0.24	45	<0.01	45	0.68
N2	20	50	51	9.2	52	8.14	53	<0.01	54	3.32
N3	30	50	52	2.6	52	39.5	52	<0.01	51	6.18
N4	40	80	83	12.6	83	84	83	<0.01	83	13.09
N5	50	100	106	52.3	106	228	105	0.01	103	33.01
N6	60	100	103	261	103	310	103	0.01	102	50.12
N7	70	100	106	671	106	554	107	0.01	104	57.04
N8	80	80	85	1142	85	810	84	0.01	82	31.36
N9	100	150	155	4431	155	1715	152	0.01	152	185.94
N10	200	150	154	2×10^4	154	6066	152	0.02	151	1154.18
N11	300	150	155	8×10^4	155	3×10^4	152	0.03	151	3763.17
N12	500	300	313	4×10^5	312	6×10^4	306	0.06	304	5864.27
Average Gap(%)	-	-	3.72	-	3.85	-	4.35	-	3.46	-

References

1 Lodi A, Martello S, Monaci M. Two-dimensional packing problems: a survey. *European Journal of Operational Research*, 2002, **141**(2): 241∼252

2 Dowsland K A, Dowsland W B. Packing problems. *European Journal of Operational Research*, 1992, **56**(1): 2∼14

3 Pisinger D. Heuristics for the container loading problem. *European Journal of Operational Research*, 2002, **141**(2): 382∼392

4 Hochbaum D S, Wolfgang M. Approximation schemes for covering and packing problems in image processing and VLSI. *Journal of the Association for Computing Machinery*, 1985, **32**(1): 130∼136

5 Leung J Y, Tam T W, Wong C S, Young G H, Chin F Y L. Packing squares into a square. *Journal of Parallel and Distributed Computing*, 1990, **10**(3): 271∼275

6 Beasley J E. An exact two-dimensional non-guillotine cutting tree search procedure. *Operations Research*, 1985, **33**(1): 49∼64

7 Hadjiconstantinou E, Christofides N. An exact algorithm for the orthogonal 2D cutting problems using guillotine cuts. *European Journal of Operational Research*, 1995, **83**(1): 21∼38

8 Gilmore P C, Gomory R E. A linear programming approach to the cutting stock problem (part I). *Operational Research*, 1961, **9**: 849∼859

9 Christofides N, Whitlock C. An algorithm for two-dimensional cutting problems. *Operational Research*, 1977, **25**(1): 30∼44

10 Cung V D, Hifi M, Cun B C. Constrained two-dimensional cutting stock problems a best-first branch-and-bound algorithm. *International Transactions in Operational Research*, 2000, **7**(3): 185∼210

11 Zhang De-Fu, Li Xin. A personified annealing algorithm for circles packing problem. *Acta Automatica Sinica*, 2005, **31**(4): 590∼595 (in Chinese)

12 Chazelle B. The bottom-left bin packing heuristic: an efficient implementation. *IEEE Transactions on Computers*, 1983, **32**(8): 697∼707

13 Berkey J, Wang P. Two-dimensional finite bin packing algorithms. *Journal of the Operational Research Society*, 1987, **38**: 423∼429

14 Kroger B. Guillotineable bin packing: a genetic approach. *European Journal of Operational Research*, 1995, **84**(3): 645∼661

15 Jakobs S. On genetic algorithms for the packing of polygons. *European Journal of Operational Research*, 1996, **88**(1): 165∼181

16 Liu D, Teng H. An improved BL-algorithm for genetic algorithm of the orthogonal packing of rectangles. *European Journal of Operational Research*, 1999, **112**(2): 413∼419

17 Dagli C, Poshyanonda P. New approaches to nesting rectangular patterns. *Journal of Intelligent Manufacturing*, 1997, **8**(3): 177∼190

18 Hopper E, Turton B. A genetic algorithm for a 2D industrial packing problem. *Computers and Industrial Engineering*, 1999, **37**(1): 375∼378

19 Hopper E, Turton B. An empirical investigation of meta-heuristic and heuristic algorithms for a 2D packing problem. *European Journal of Operational Research*, 2001, **128**(1): 34∼57

20 Wu Y L, Huang W Q, Lau S C, Wong C K, Young G H. An effective quasi-human based heuristic for solving the rectangle packing problem. *European Journal of Operational Research*, 2002, **141**(2): 341∼358

21 Hifi M, Hallah R. A best-local position procedure-based heuristic for the two-dimensional layout problem. *Studia Informatica Universalis, International Journal on Informatics–Special Issue on Cutting, Packing and Knapsacking Problems*, 2002, **2**(1): 33∼56

22 Hifi M, Hallah R. A hybrid algorithm for the two-dimensional layout problem: the cases of regular and irregular shapes. *International Transactions in Operational Research*, 2003, **10**(3): 195∼216

23 Burke E, Kendall G, Whitwell G. A new placement heuristic for the orthogonal stock-cutting problem. *Operations Research*, 2004, **52**(4): 655∼671

24 Zhang D F, Kang Y, Deng S. A new heuristic recursive algorithm for the strip rectangular packing problem. *Computers & Operations Research*, 2006, **33**(8): 2209∼2217

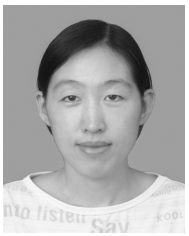
25 Zhang D F, Deng A S, Kang Y. A hybrid heuristic algorithm for the rectangular packing problem. *Lecture Notes in Computer Science*, 2005, **3514**: 783∼791

26 Zhang D F, Liu Y J, Chen S D, Xie X G. A meta-heuristic algorithm for the strip rectangular packing problem. *Lecture Notes in Computer Science*, 2005, **3612**: 1235∼1241

27 Davis L. *Handbook of Genetic Algorithms*. New York: Van Nostrand Reinhold, 1991

ZHANG De-Fu Associate professor at School of Information Science and Technology, Xiamen University. He received his bachelor and master degrees in computational mathematics from Xiangtan University in 1996 and 1999, respectively, and his Ph. D. degree in computer software and its theory from Huazhong University of Science and Technology in 2002. His research interest covers computational intelligence and financial data mining. Corresponding author of this paper. E-mail: dfzhangl@xmu.edu.cn

CHEN Sheng-Da Master student at Xiamen University. He received his bachelor degree from Jimei University in 2004. His research interest is computational intelligence. E-mail: cshengda@126.com



LIU Yan-Juan Master student in Xiamen University. She received her bachelor degree from Shijiazhuang University of Economics in 2004. Her research interest is computational intelligence.
E-mail: jjj514@sohu.com