A Novel Greedy Computing Algorithm for Rectangle Packing Problems

Yanbing LIU[1,2] , Duanbing CHEN[3]

[1] Chongqing University of Posts and Telecommunications,Chongqing 400065, P. R. China
[2]School of Computer Science, UEST of China, Chengdu 610054,P. R. China
[3] College of Computer Science, Huazhong University of Science and Technology, Wuhan 430074, China

Summary
Rectangle packing problem often appears in encasement and cutting as well as layout of homepage or newspaper, etc. This problem has been shown to be NP hard. For solving this problem, many compute algorithms based on different strategies are presented in the literatures. A novel-computing algorithm is proposed in this paper. The novel match algorithm tested the instances that taken from the literatures. The computational results demonstrate that the novel algorithm is rather optimal and efficient for solving rectangle block packing problem.

Key words:
Rectangle packing; packing; computing algorithm; matching degree.

Introduction

Rectangle block packing problem often appears in encasement and cutting as well as layout of newspaper or homepage, etc. This problem belongs to a subset of classical cutting and packing problems and has been shown to be NP hard. For more extensive and detailed descriptions of packing problem, please refer to Lodi (2002)[1]and Pisinger (2002)[2]. For solving this problem, various algorithms based on different data structures have been suggested, for example, SP [3], TCG [4] and CBL [5]. In order to improve the performance of the algorithm, some literatures combine genetic algorithm or simulated annealing with deterministic method and obtain hybrid algorithm[6]. Recently, some robust heuristic algorithms are presented [7][8]. These two heuristic algorithms are fast and effective. Some people formalize the experience and wisdom of human being and obtain the quasi-human heuristic algorithm[9].

In this paper, a greedy-computing algorithm (GCA) for rectangle block packing problem is proposed. The objective is to minimize the dead space of the box. The key point of this algorithm is that the rectangle block packed into the box always matches the blank place, and the matching degree should be as large as possible. In this way, the blocks will be close to each other wisely, and the spare space is decreased. As compared with literatures, the results from MCA are much improved.

For instances ami33_R and ami49_R, the dead space obtained by MCA are 4.82% and 4.58%, respectively, whereas obtained by algorithm in [10] are 6.83% and 5.52%, respectively. For instances test1 and test2 provided by Lin (2002)[4], we obtain the optimal solutions. Computational results show that the MCA is rather efficient for solving rectangle block packing problem.

The remaining of this paper is organized as follows. Section 2 discusses problem descriptions and framework of our method. Section 3 is computing procedure of the algorithm. Computational results are presented in Section 4. Conclusions are drawn in Section 5.

2. Problem Descriptions and Framework of Our Method

A rectangular container B_0 of width w_0 and height h_0 is given. And n rectangle blocks B_1, B_2,…, B_n of deterministic shape and size are given. In the plane rectangular coordinates, the bottom left of the container is placed at (0, 0) with its four sides parallel to X- and Y- axis, respectively. The objective is to pack as many rectangle blocks into the container B_0 as possible, that is, to minimize the dead space of the container. The constraints for packing rectangle blocks are:

-Each edge of a rectangle block must be parallel to X-axis or Y-axis.

-There is no overlapping for any two rectangle blocks, i.e., the overlapping area is zero.

2.1 Fundamental Conceptions

1) Matching movement (MM)

A packing movement is called a *matching movement* (MM), if the edges of the rectangle to be packed overlap the different directional edges with other already packed rectangles including the box, and the overlapping lengths are longer than zero. E.g., in Fig. 1, the shadowy rectangle blocks have been packed, and the rectangle block d is

outside the box. The packing movement is a MM, if block *d* is situated at place *A*, *B* or *C*; it is not a MM if situated at place *D* or *E*.

In particular, a packing movement is called a *perfect matching movement* if the rectangle block to be packed not only matches a blank place, but also touches some other previously packed rectangle blocks including the box. For example, in Fig. 1, if rectangle block *d* is packed at place *A*, it matches the blank place formed by rectangle block *a* and *b*, furthermore, it touches the rectangle block *c*. Thus, the movement of packing rectangle block *d* is a perfect matching movement.

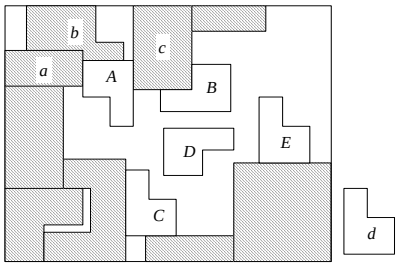


Fig. 1 Matching movement

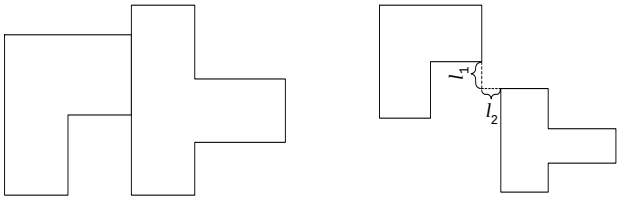


Fig. 2 Distance between two rectangles

2) Distance between two rectangle blocks

For two given rectangle blocks B_i and B_j, let point P_i belonging to B_i, and P_j belonging to B_j. Let the distance between P_i and P_j be $d(P_i, P_j)$. The distance between B_i and B_j is defined as $d_{ij} = \min_{P_i \in B_i, P_j \in B_j} d(P_i, P_j)$.

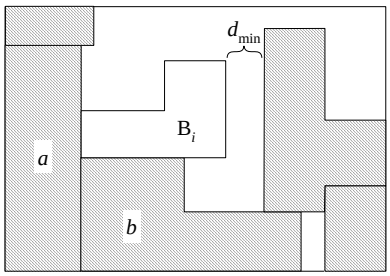


Fig. 3 Matching degree

For example, (a) to (b) as shown in Fig. 2, the distance between two rectangle blocks is zero and l_1+l_2, respectively.

3) Distance between one rectangle block and several rectangle blocks

For a given rectangle block *B* and a set of rectangle blocks $\{B_i | i=1,2,\ldots, m\}$. Let the distance between *B* and B_i (i=1,2,…, m) be d_i. The minimum of d_i (i=1,2,…, m) is defined as the distance between *B* and *m* rectangle blocks $B_1, B_2,\ldots, B_m$.

4) Matching degree of MM

As shown in Fig. 3, if a rectangle block B_i is packed into the box according to a MM, and let the distance between rectangle block B_i and all the already packed rectangle blocks including the box (except the rectangle blocks that form this blank place) be $d_{\min}$, the matching degree MD_i of the corresponding MM can be defined as:

$$MD_i = 1 - \frac{d_{\min}}{\sqrt{S_i}} \quad (1)$$

Where S_i is the area of B_i.

2.2 Main Idea and Sketch of MCA Algorithm

If some rectangle blocks have been packed into the box without overlapping. The question is which one is the best candidate for the remainder, and which position is the best one to be filled? According to the packing experience in our daily life, we can find a match for a rectangle block and pack it. So, we always pack a rectangle block according to the following principle: The rectangle to be packed into the box always matches a blank space, and the matching degree should be as large as possible. At each step, we do the MM with the largest matching degree until no rectangle block is left outside the box or no rectangle block can be packed according to the current configuration. In order to improve the performance, we introduce the backtracking process.

3. Computing Procedure of MCA Algorithm

This paper makes the elucidation of the algorithm to compute and solve rectangle block packing problem.

First stage is greedy computing procedure.

Step 1. Based on the current configuration, if there is no rectangle block can be packed, return the dead space of the box and terminate the program; otherwise, enumerate all MMs, and calculate the matching degree for each MM.

Step 2. Select the MM with the largest cave degree and pack the corresponding rectangle block. A new configuration is obtained.

Step 3. If all rectangle blocks are packed into the box, output the packing result and stop successfully. Otherwise, return to step 1.

Second stage is running the backtracking procedure.

Step 1. Based on the current configuration, if there is no rectangle block can be packed, output the dead space of the box and terminate the program; otherwise, enumerate all MMs as candidates

Step 2. For each candidate MM, pseudo-pack (just pack the rectangle block into the box temporarily, it will be removed from the box future) corresponding rectangle block and get a new configuration. Based on this new configuration, pseudo-pack the remainder rectangle blocks according to the *greedy computing procedure*. If all rectangle blocks can be packed, output the packing result and stop successfully. Otherwise, calculate the dead space of the box according to the tentative end configuration as the penalty score of corresponding candidate MM.

Step 3. Select the MM with the lowest penalty score. If there is just one MM with the lowest penalty score, pick this one and pack the corresponding rectangle block. Otherwise, select the MM with the largest matching degree and pack the corresponding rectangle block. A new configuration is obtained, return to step 1.

Table 1: Computational results

				Our method		Yang et al (2004)		Lin et al (2002)	
Instance	# Total blocks	#Rectangular blocks	# Rectangle blocks	Dead space %	Time Sec	Dead space %	Time Sec	Dead space %	Time Sec
Ami33_R	39	33	6	4.82	386.99	6.83	93.17		
Ami49_R	55	49	6	4.58	1251.29	5.52	246.38		
Test1	17	6	11	0.00	2.41			9.375	1224
Test2	29	22	7	0.00	1.12			6.944	1409

4 Computational Results

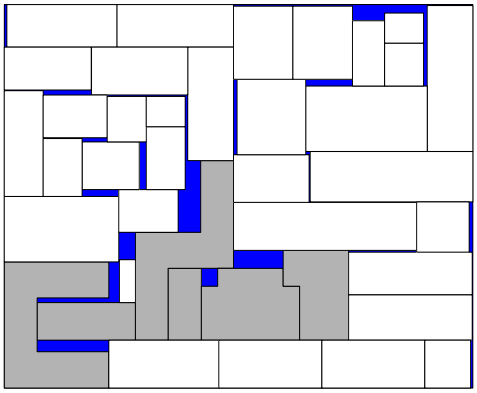
The algorithm MCA is implemented by C#.net programming language. Performance of the MCA has been tested with four instances taken from [4] and [10]. Instances ami33_R and ami49_R are taken from Yang (2004)[10]. There are 33 rectangular blocks and 6 rectangle blocks for ami33_R, 49 rectangular blocks and 6 rectangle blocks for ami49_R, as shown in Table 1. Instance test1 and test2 are taken from [4], and the optimal solutions for these two instances are known. The details of test1 and test2 please refer to Lin (2002)[4]. The layouts of ami33_R, ami49_R, test1 and test2 are shown in Fig. 4. The comparisons between [4], [10] and our method are shown in Table 1. For instances ami33_R and ami49_R, the dead space obtained by our method is 4.82% and 4.58%, and the runtime is 386.99 and 1251.29 seconds, respectively. Whereas, the dead space obtained by Yang (2004)[10] is 6.83% and 5.52% with the runtime is 93.17 and 246.38 seconds, respectively. For instances test1 and test2, the optimal solutions can be obtained by our method, that is, the dead space is 0%. In the Lin (2002)[4], the dead space is 9.375% and 6.944%, respectively. The runtime used by our method is shorter than that by Lin (2002)[4], as shown in last two rows of Table 1.

5 Conclusions

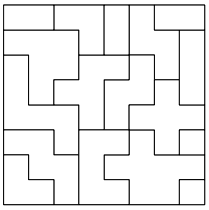
In this paper, an optimal computing algorithm for rectangle block packing problem is proposed. This algorithm within reasonable runtime can obtain low dead space of the box. For instance ami33_R and ami49_R taken from Yang (2004)[10], the dead space obtained by our method is 4.82% and 4.58%, respectively; and for instance test-1 and test-2 taken from Lin (2002)[4], the optimal solutions are obtained. The computational results demonstrate that the optimal algorithm proposed in this paper is rather efficient for solving rectangle block packing problem.

Acknowledgments

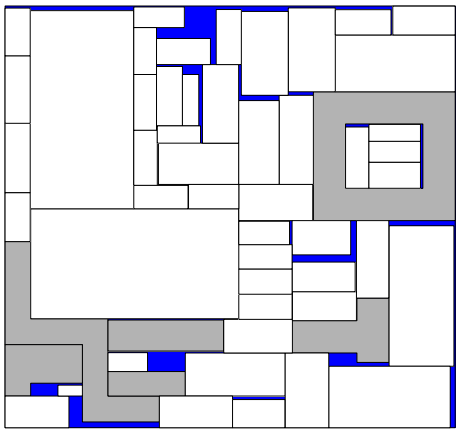
The authors would like to thank Dr. Y.Y. Yao and Dr. J.T. Yao for their help during my visit in Canada. We also wish to express my sincere thanks to all those who have worked or are currently working with me for their helpful discussions. The Natural Science Foundation of CQMEC under Grant No.KJ050507, and the Natural Science Foundation of CQUPT, CSTC under Grant No.2005BB2060 supported the work.



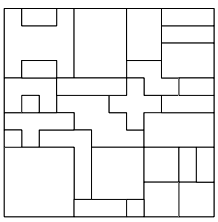
Instance: Ami33_R
Box dimensions: 1433x1067
Dead space: 4.82%



Instance: Test1
Box dimensions: 8x8
Dead space: 0.00%



Instance: Ami49_R
Box dimensions: 6990x6550
Dead space: 4.58%



Instance: Test2
Box dimensions: 12x12
Dead space: 0.00%

Fig. 4 The layouts of instance ami33_R, ami49_R, test1 and test2

References

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

[2] Pisinger D(2002). Heuristics for the container loading problem. European Journal of Operational Research 141: 382-392.

[3] Kang M Z and Dai W W-M (1998). Arbitrary rectangle block packing based on sequence pair. ICCAD98, San Jose, CA, USA, 259-266.

[4] Lin J-M, Chen H-L and Chang Y-W (2002). Arbitrary shaped rectangle module placement using the transitive closure graph representation. IEEE transaction on Very Large Scale Integration System 10(6): 886-901.

[5] Ma Y, et al (2003). Arbitrary convex and concave rectangle block packing based on corner block list. IEEE 2003, V493-496.

[6] Leung T W, Chan C K and Troutt M D (2003). Application of a mixed simulated annealing-genetic algorithm heuristic for the two-dimensional orthogonal packing problem. European Journal of Operational Research 145: 530-542.

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

[8] Zhang D, Deng A and Kang Y (2005). A hybrid heuristic algorithm for the rectangular packing problem. Lecture Notes on Computer Science (ICCS 2005) 3514: 783-791.

[9] Huang W, Li Y, Akeb H and Li C (2005). Greedy algorithms for packing unequal circles into a rectangular container. Journal of the Operational Research Society 56: 539-548.

[10] Yang Z, Dong S, Hong X and Wu Y-L (2004). Arbitrary rectangle block packing based on less flexibility first principles. Chinese Journal of Semiconductors 25(11): 1416-1422.

Yanbing LIU received M.S. degree in computer application from Beijing University of Posts and Telecommunications, Beijing, China, in 2001. He is working toward the Ph.D. degree at University of Electronic Science and Technology of China. Since 2002, he has been an Associate Professor at the Chongqing University of Posts and Telecommunications. He also serves as a teacher of CISCO Network Technology Academy. His current research interests lie in the study of the evaluation and planning of the capacity and performance of wireless networks, traffic analysis of wireless, traffic modeling, resource assignment and resource allocation.