Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computers & Operations Research 33 (2006) 2209–2217
computers & operations research
www.elsevier.com/locate/cor

A new heuristic recursive algorithm for the strip rectangular packing problem

Defu Zhang[a,*], Yan Kang[b], Ansheng Deng[a]

[a] *Department of Computer Science, Xiamen University, 361005, China*
[b] *School of Software, Yunnan University, Kunming 650091, China*

Available online 26 February 2005

Abstract

A fast and new heuristic recursive algorithm to find a minimum height for two-dimensional strip rectangular packing problem is presented. This algorithm is mainly based on heuristic strategies and a recursive structure, and its average running time is $T(n) = \theta(n^3)$. The computational results on a class of benchmark problems have shown that this algorithm not only finds shorter height than the known meta-heuristic ones, but also runs in shorter time. Especially for large test problems, it performs better.

Keywords: Packing problems; Recursion; Heuristic

1. Introduction

Packing problems have found many industrial applications, with different applications incorporating different constraints and objects. 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 newspaper paging, articles and advertisements have to be arranged in pages. In the shipping industry, a batch of objects of various sizes have to be shipped as many as possible in a larger container, and a bunch of optical fibers have to be accommodated in a pipe with perimeter as small as possible. In VLSI floor planning, VLSI has to be laid. These applications can formalize as packing problems [1]. For more extensive and detailed descriptions of packing problems, the reader is referred to [1–3].

* Corresponding author. Tel.: +86 0592 2187472; fax: +86 0592 2183502.
E-mail address: dfzhang@xmu.edu.cn (D. Zhang).

0305-0548/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2005.01.009

In this paper, a two-dimensional strip rectangular packing problem is considered and its object is to find the minimum height. This problem belongs to a subset of classical cutting and packing problems and has been shown to be NP hard [4,5]. Optimal algorithms for orthogonal two-dimensional cutting were proposed in [6,7]. However, they might not be practical for large problems. Some heuristic algorithms were developed by Berkey and Wang [8]. Hybrid algorithms combining genetic and deterministic methods for the orthogonal packing problem were proposed by Jakobs [9], Liu and Teng [10], and Dagli and Poshyanonda [11]. An empirical investigation of meta-heuristic and heuristic algorithms of the strip rectangular packing problems was given by Hopper and Turton [12]. An effective quasi-human heuristic, Less Flexibility First, for solving the rectangular packing problem was presented by Wu et al. [13]. However, generally speaking, those non-deterministic algorithms are more time consuming and are hence less practical for problems having a large number of rectangles. Recently, some new models and algorithms were developed by Belov [14], Caprara and Monaci [15], and Fekete and Schepers [16]. In this paper, we will present a rather fast and effective heuristic recursive algorithm for solving the orthogonal strip rectangular packing problem. The computational results on a class of benchmark problems show that this algorithm not only finds shorter height than the known meta-heuristic ones but also runs in shorter time. Especially for large test problems, it performs better.

The rest of this paper is organized as follows. In Section 2, we give a clear mathematical formulation for the strip rectangular packing problem. In Section 3, based on the presented heuristic strategies, we present a heuristic recursive algorithm. Computational results are described in Section 4. Conclusions are summarized in Section 5.

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.

Given a rectangular board with given width W, and n rectangles with length l_i and width w_i, $1 \leqslant i \leqslant n$, take the origin of two-dimensional Cartesian coordinate system at the bottom-left corner of the rectangular board, (x_{L}, h) denotes the top-left corner coordinates of the rectangular board and $(x_{\mathrm{R}}, y_{\mathrm{R}})$ denotes the bottom-right corner coordinates of this board (see Fig. 1). The aim of this problem is to find a solution composed of n sets of quadruples

$$P = \{\langle (x_{\ell \mathrm{i}}, y_{\ell \mathrm{i}}), (x_{\mathrm{ri}}, y_{\mathrm{ri}})\rangle \mid 1 \leqslant i \leqslant n, x_{\ell \mathrm{i}} < x_{\mathrm{ri}}, y_{\ell \mathrm{i}} > y_{\mathrm{ri}}\},$$

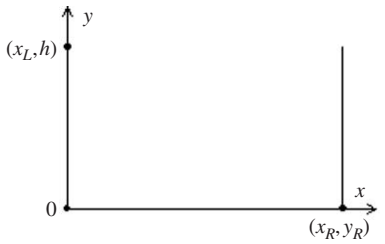

Fig. 1.

where $(x_{\ell i}, y_{\ell i})$ denotes the top-left corner coordinates of rectangle i, and (x_{ri}, y_{ri}) denotes the bottom-right corner coordinates of rectangle i. For all $1 \leqslant i \leqslant n$, the coordinates of rectangle i satisfies the following conditions:

(1) $x_{ri} - x_{\ell i} = l_i \wedge y_{\ell i} - y_{ri} = w_i$ or $x_{ri} - x_{\ell i} = w_i \wedge y_{\ell i} - y_{ri} = l_i$.
(2) For all $1 \leqslant j \leqslant n$, $j \neq i$, rectangle i and j cannot overlap, namely
$x_{ri} \leqslant x_{\ell j}$ or $x_{\ell i} \geqslant x_{rj}$ or $y_{ri} \geqslant y_{\ell j}$ or $y_{\ell i} \leqslant y_{rj}$.
(3) $x_L \leqslant x_{\ell i} \leqslant x_R$, $x_L \leqslant x_{ri} \leqslant x_R$ and $y_R \leqslant y_{\ell i} \leqslant h$, $y_R \leqslant y_{ri} \leqslant h$,

such that the used board height h is minimized.

It is noted that the orthogonal rectangular packing problems denote that the packing process has to ensure the edges of each rectangle are parallel to the x- and y-axis, respectively, namely all rectangles to be packed cannot be packed aslant.

3. Heuristic recursive (HR) algorithm

Many useful algorithms have recursive structure: to solve a given problem, they call themselves recursively one or more times to deal with closely related subproblems, so these algorithms are simple and effective. These algorithms typically follow a divide-and-conquer approach: they 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 divide-and-conquer paradigm involves three steps at each level of the recursion [17]:

(1) Divide the problem into a number of subproblems.
(2) Conquer the subproblems by solving them recursively. If the subproblem sizes are small enough, however, just solve the subproblems in a straightforward manner.
(3) Combine the solutions to the subproblems into the solution for the original problem.

Intuitively, we can construct a recursive algorithm for the strip rectangular packing problem as follows:

(1) Pack a rectangle into the space to be packed. Divide the unpacked space into two subspaces (see Fig. 2).
(2) Pack each subspace by packing them recursively. If the subspace sizes are small enough to only pack a rectangle, however, just pack this rectangle into the subspace in a straightforward manner.
(3) Combine the solutions to the subproblems into the solution for the rectangular packing problem.

In Fig. 2, there are two cases when dividing. (a) Denotes space S can be divided into a unbounded space S_1 and a bounded space S_2 after a rectangle is packed into S, here, S_1 is similar to S and is unbounded space. (b) Denotes space S_2 can be divided into S_3 and S_4 after a rectangle is packed into S_2, here, S_3 and S_4 are similar to S_2 and are bounded space. So, the recursive packing process is called to pack S_2. In detail, the recursive packing procedure can be stated as follows (see Fig. 3): During the process of packing bounded subspace S_2, this recursive procedure is used. As S_3 and S_4 are similar to S_2, RecursivePacking (S_2) can be called recursively. For this recursive procedure, all the operations in RecursivePacking() can be done in constant time. This gives an overall time complexity of $T(n) = \theta(n)$.

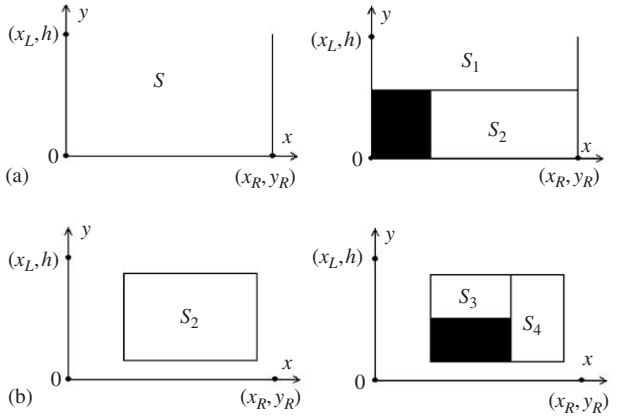

Fig. 2. Divide the unpacked space into two subspaces.

```
RecursivePacking(S2)
    if no rectangle can be packed into the bounded space S2
    then return;
     else
       Select a rectangle and pack it into S2 ;
       Divide the unpacked space into two bounded space S3 and S4 ;
       if the area of S3 > the area of S4
          S2 = S3 ; RecursivePacking(S2);
          S2 = S4 ; RecursivePacking(S2);
       else
          S2 = S4 ; RecursivePacking(S2);
          S2 = S3 ;RecursivePacking(S2);
```

Fig. 3. Recursive packing procedure for the bounded space.

It is noted that how to select a rectangle to be packed into S is very important to improve the performance of recursive algorithm. In this paper, we present a heuristic strategy for selecting a rectangle to be packed, namely a rectangle with the maximum area is given priority to pack. In detail, unpacked rectangles should be sorted by non-increasing ordering of area size. The rectangle with maximum area should be selected to pack if it can be packed into the unpacked subspace.

Packing( ) can be called when packing the unbounded space, and this procedure can be stated as follows (see Fig. 4): This packing procedure shows that the recursive procedure RecursivePacking(S_2) can be called iteratively while packing process is not finished. In order to minimize the height of packing, a heuristic packing strategy is presented as follows. The long side of rectangle to be packed should be packed along the bottom side of subspace S.

```
Packing( S )
    While packing process is not finished
        Select a rectangle and pack it by heuristic strategy into S ;
        Divide the unpacked space into the unbounded space S1 and the bounded space S2 ;
        S = S1 ;
        RecursivePacking( S2 );
```

Fig. 4. Packing procedure.

```
HeuristicRecursion()
    Sort unpacked rectangles by non-increasing ordering of area size
    for i=1 to n
      for j=i to n
          Swap the order of rectangle i and j in current orderings;
          Packing( S );
          Save the best orderings of rectangles and the best packed area so far;
     Current orderings =best orderings
```

Fig. 5. The heuristic recursive (HR) algorithm.

Since the packing orderings affect the performance of the recursive algorithm, several orderings can be tried to enhance the performance. In practice, if we strictly pack rectangles in order of their area, the way of this kind of packing does not correspond to practical packing in industry fields. Therefore, we can try different orderings to try and find a better solution.

In detail, the HR algorithm can be stated as follows (see Fig. 5): From the description of heuristic recursive algorithm, we can know the average running time of this algorithm is $T(n) = \theta(n^3)$. It is noted that, in order to enhance the performance of the recursive algorithm, HeuristicRecursion() can be run repeatedly until h cannot improve.

4. Computational results

Performance of the HR algorithm has been tested with seven different sized test instances ranging from 16 to 197 items [12] (see Table 1). The optimal solutions of these test instances are all known. In order to compare HR with known meta-heuristic, two best meta-heuristic GA+BLF and SA+BLF in [12] are selected, they are run on a PC with a Pentium Pro 200 MHz processor and 65 MB of RAM under Windows NT4.0 [12], so the relative distance of best solution to optimum height (RDBSOH) and the running time are directly taken from [12]. The results obtained from the realization of the GA+BLF and SA+BLF are also given. Our experiments were run on a Dell GX260 with a 2.4 GHz CPU. Dell GX260 with a 2.4 GHz CPU is about 13 times faster than PC with a Pentium Pro 200 MHz processor, therefore we adjust the running time of meta-heuristic by dividing by 13. The computational results are reported in Tables 2 and 3. In order to show performance comparisons more clearly, we give the relative distance

Table 1
Test problems

Problem category	Number of items: n	Optimal height	Object dimensions
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
RDBSOH of GA+BLF, SA+BLF and HR (%)

	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
HR	8.33	4.45	6.67	2.22	1.85	2.5	1.8	3.97

Table 3
Average running time of GA+BLF, SA+BLF and HR (s)

	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	152.13	530.15	1761.02	19274.41	3107.2
HR	0	0	0.03	0.14	0.69	2.21	36.07	5.59

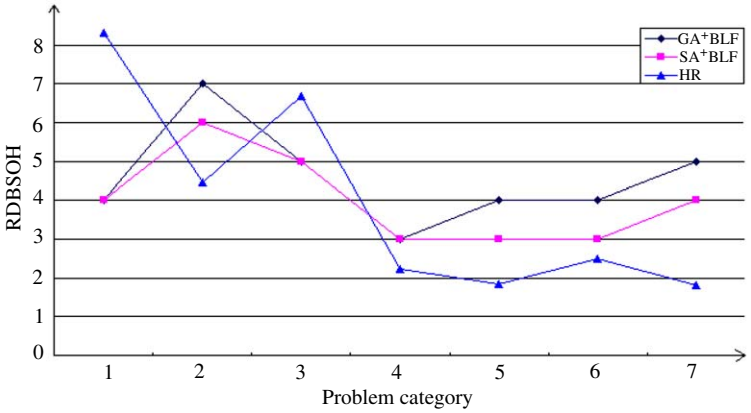

Fig. 6. RDBSOH of GA+BLF, SA+BLF and HR (%).

of best solution to optimum height (%) comparison in Fig. 6 and the running time comparison in Fig. 7. In addition, we give four packed results on test instances C11, C12, C13 and C72 for HR in Figs. 8 and 9. Here, the number in the right of Figs. 8 and 9 is optimal height.

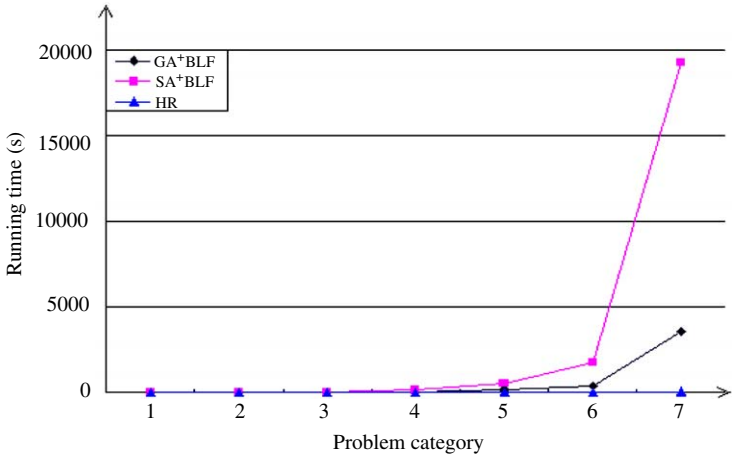

Fig. 7. Average running times of GA+BLF, SA+BLF and HR (s).

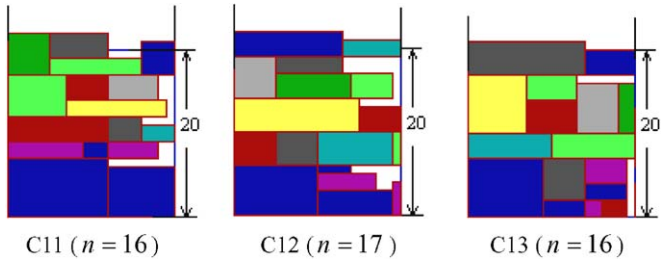

Fig. 8. Packed results of C1 for HR.

On this test set, as shown in Table 2, RDBSOH of HR ranges from 1.8% to 8.33% with the average RDBSOH 3.97%. The average RDBSOH of GA+BLF and SA+BLF are 4.57 and 4, respectively. The average RDBSOH of HR is lower than that of GA+BLF and SA+BLF. From Fig. 6, we can observe that for C2 and C4–C7, RDBSOH of HR is lower than that for GA+BLF and SA+BLF. What is more, RDBSOH of HR decreases as the number of rectangles increases. As shown in Table 3 and Fig. 7, the computational speed of HR is faster than that of GA+BLF and SA+BLF for all test instances. In particular, HR performs better than GA+BLF and SA+BLF for the bigger test instances, C6 and C7.

5. Conclusions

A new heuristic recursive (HR) algorithm for the orthogonal rectangular packing is presented in this paper. This algorithm is very simple and intuitive, and can solve the rectangular packing problem fast. The computational results have shown that the HR algorithm outperforms the known meta-heuristic ones in relative distance of best solution to optimum height and the running time. Especially for large test problems, it performs better. What is more, HR has no parameters. However, meta-heuristics involve in many parameters, their performance depends on the parameters selection. So HR may be of great practical

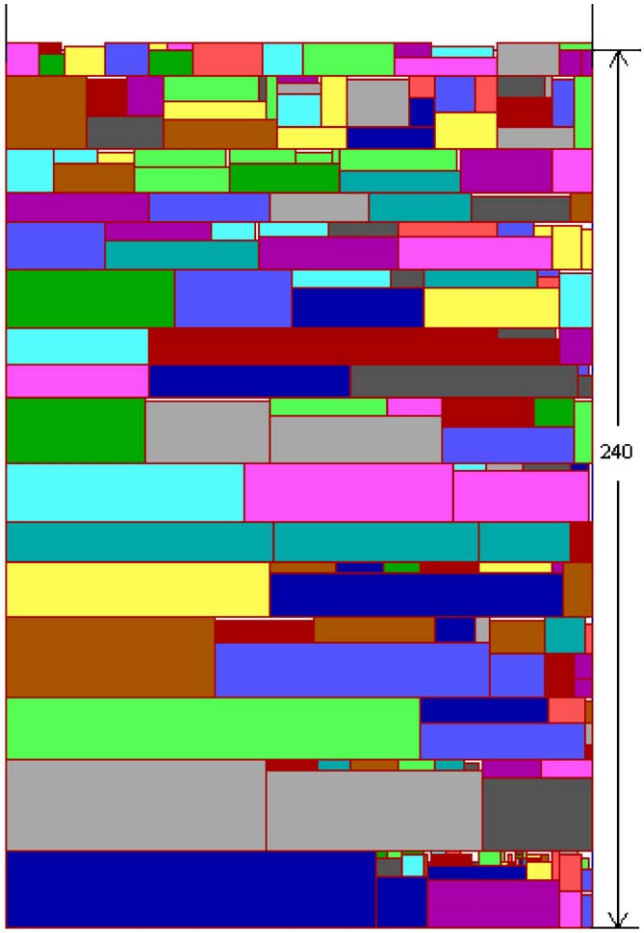

Fig. 9. Packed result of C72 ($n = 197$) for HR.

value to the rational layout of the rectangular objects in the engineering fields, such as the wood, glass and paper industry, and the ship-building industry, textile and leather industry. Future work is to further improve the performance of the HR algorithm and extend it to three-dimensional rectangular packing problems.

Acknowledgements

The authors would like to thank the referees for their helpful comments and suggestions that help to improve this paper. This research has been supported by academician start-up fund (Grant No. X01122) in Xiamen University.

References

[1] Lodi A, Martello S, Monaci M. Two-dimensional packing problems: a survey. European Journal of Operational Research 2002;141:241–52.
[2] Dowsland KA, Dowsland WB. Packing problems. European Journal of Operational Research 1992;56:2–14.
[3] Pisinger D. Heuristics for the container loading problem. European Journal of Operational Research 2002;141:382–92.
[4] Hochbaum DS, Wolfgang M. Approximation schemes for covering and packing problems in image processing and VLSI. Journal of the Association for Computing Machinery 1985;32(1):130–6.
[5] Leung J, Tam T, Wong CS, Young G, Chin F. Packing squares into square. Journal of Parallel and Distributed Computing 1990;10:271–5.
[6] Beasley JE. An exact two-dimensional non-guillotine cutting tree search procedure. Operations Research 1985;33:49–64.
[7] Hadjiconstantinou E, Christofides N. An optimal algorithm for general orthogonal 2-D cutting problems. Technical report MS-91/2, Imperial College, London, UK.
[8] Berkey JO, Wang PY. Two-dimensional finite bin packing algorithms. Journal of the Operational Research Society 1987;38:423–9.
[9] Jakobs S. On genetic algorithms for the packing of polygons. European Journal of Operational Research 1996;88:165–81.
[10] Liu D, Teng H. An improved BL-algorithm for genetic algorithm of the orthogonal packing of rectangles. European Journal of Operational Research 1999;112:413–9.
[11] Dagli CH, Poshyanonda P. New approaches to nesting rectangular patterns. Journal of Intelligent Manufacturing 1997;8:177–90.
[12] Hopper E, Turton BCH. An empirical investigation of meta-heuristic and heuristic algorithms for a 2D packing problem. European Journal of Operational Research 2001;128:34–57.
[13] Wu YL, Huang W, Lau SC, Wong CK, Young GH. An effective quasi-human based heuristic for solving the rectangle packing problem. European Journal of Operational Research 2002;141:341–58.
[14] Belov G. Problems, models and algorithms in one- and two-dimensional cutting. PhD thesis, Technischen Universitat Dresden, 2003. http://www.math.tu-dresden.de/belov/publ/text030908_SUBMIT.pdf.
[15] Caprara A, Monaci M. On the two-dimensional Knapsack Problem. Operations Research Letters 2004;32:5–14.
[16] Fekete SP, Schepers J. On higher-dimensional packing III: exact algorithms. Operations Research 2004; Technical Report ZPR 97-290. http://www.zaik.uni-koeln.de/~paper/preprints.html?show = zpr97-290&preprint_session = 9b65 7999e2faecfc37a9ce6acf104144.
[17] Cormen TH, Leiserson CE, Rivest RL, Stein C. Introduction to algorithms. 2nd ed., Cambridge, MA: The MIT Press; 2001.