Expert Systems with Applications 40 (2013) 5542–5550

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Heuristics for two-dimensional strip packing problem with 90° rotations

Kun He, Yan Jin *, Wenqi Huang

School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China

ARTICLE INFO

Keywords:
Combinatorial optimization
Strip packing
Heuristics
Deterministic algorithm

ABSTRACT

This paper proposes a deterministic heuristic algorithm (DHA) for two-dimensional strip packing problem where 90° rotations of pieces are allowed and there is no guillotine packing constraint. The objective is to place all pieces without overlapping into a strip of given width so as to minimize the total height of the pieces. Based on the definition of action space, a new sorting rule for candidate placements is proposed such that the position for the current piece is as low as possible, the distance between the current piece and other inside pieces is as close as possible, and the adverse impact for further placements is as little as possible. Experiments on four groups of benchmarks showed the proposed DHA achieved highly competitive results in comparison with the state-of-the-art algorithms in the literature. Also, as a deterministic algorithm, the DHA could achieve high quality solutions by only one independent run on both small-scale and large-scale problem instances and the results are repeatable.

1. Introduction

The two-dimensional strip packing problem (2D-SPP) is an open-dimension-problem in which a set of rectangular pieces have to be orthogonally placed into a rectangular strip with fixed width and infinite height to minimize the total height of the pieces. This problem is a significant NP-hard problem with many operational applications in various industries, including the wood, glass, metal industries, which aims to find very good solutions within reasonable times. To meet different manufacturing needs of industries, the orientation constraint and the guillotine constraint may be considered. The orientation constraint includes two cases: the orientation of all pieces is fixed or the pieces could be rotated by 90° when they are placed into the strip. With the guillotine issue, there are two versions that the pieces are guillotine packing or non-guillotine packing, the detailed information of which may refer to Christofides and Whitlock (1977). In this paper, we assume that the pieces could be rotated by 90° and consider non-guillotine packing.

The 2D-SPP is such a challenging problem that has been extensively studied by the researchers in recent decades. The algorithms proposed in the literature can be divided into two categories: the exact method and the heuristic one.

The researchers proposed a number of heuristic algorithms to solve 2D-SPP, including the bottom-left algorithm proposed by Baker, Coffman, and Rivest (1980), the bottom-left-fill algorithm proposed by Chazelle (1983), the best-fit algorithm proposed by Burke, Kendall, and Whitwell (2004) and the flexibility first algorithm proposed by Wu, Huang, Lau, Wong, and Young (2002). Then Burke, Kendall, and Whitwell (2006) combined the best-fit (BF) heuristic with tabu search (TS), simulated annealing (SA) and genetic search (GA) respectively to propose three new algorithms BF + TS, BF + SA, and BF + GA. Zhang, Kang, and Deng (2006) proposed a heuristic recursion algorithm (HR). Huang and Chen (2007) proposed a caving degree based heuristic algorithm (HA). Alvarez-Valdes, Parreño, and Tamarit (2008) proposed a greedy randomized adaptive search procedure (GRASP). Belov, Scheithauer, and Mukhacheva (2008) proposed the SubKP and bottom-left–right (BLR) algorithms and Burke, Hyde, and Kendall (2011) proposed a squeaky wheel optimization algorithm (Squeaky wheel). Leung and Zhang (2011) proposed a fast layer-based heuristic algorithm (FH).

To our knowledge, GRASP, SVC, BF + TS, BF + SA, BF + GA and FH are the state-of-the-art heuristic algorithms in the literature. The GRASP algorithm combined the constructive phase which selected a promising piece to place and the improving phase which corrected some wrong decisions to improve the solution. The SVC algorithm filled the empty spaces by solving a one-dimensional knapsack problem and assigned suitable pseudo-profits in the widths of the pieces. BF + TS, BF + SA, BF + GA apply tabu search, simulated annealing, genetic search based on the best-fit heuristic respectively. The FH algorithm is mainly based on heuristic strategies inspired by the wall-building rule of bricklayers.

Compared with works on the heuristic method, there are only limited works on the exact one, including (Christofides & Hadjicon-

* Corresponding author. Tel.: +86 02787543885.
E-mail addresses: brooklet60@gmail.com (K. He), jinyan.hust@gmail.com (Y. Jin), wqhuangwh@gmail.com (W. Huang).

0957-4174/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2013.04.005

stantinou, 1995; Cui, Yang, Cheng, & Song, 2008; Kenmochi, Imamichi, Nonobe, Yagiura, & Nagamochi, 2009; Lesh, Marks, McMahon, & Mitzenmacher, 2004; Martello, Monaci, & Vigo, 2003). Owing to the restriction on running time and computer memory, however, these exact algorithms still can not effectively handle the problems with large-scale data sets.

To the best of our knowledge, the most efficient algorithms of the exact method are the staircase and G-staircase algorithms which were proposed by Kenmochi et al. (2009). They applied the staircase placement as the branching operation and dynamic programming as the bounding operation, in such a way they could handle most data sets with 49 rectangular pieces and several data sets with up to 500 pieces. Compared with other exact algorithms, the sizes of solved instances are much larger.

According to whether the result is repeatable, we can classify the heuristic algorithms into two categories, namely the random algorithm and the deterministic algorithm. BF + TS, BF + SA, BF + GA, HR, GRASP, SVC and BLR belong to the former while HA, FH and Squeaky wheel belong to the latter.

In this paper, based on the characteristic of the strip packing problem, we present a new deterministic heuristic algorithm (DHA) for solving the 2D-SPP. The DHA integrates a rapid constructive phase and a partial tree search phase, which holds two key ingredients together to ensure a high efficient performance. First, we define a new sorting rule for all candidate placements such that the position for the current piece is as low as possible, the distance between the current piece and other pieces already placed is as close as possible, and the adverse impact for further placements is as little as possible. Besides, we apply a partial tree search method to improve the solution. At each iteration step, we select the top N promising placements to pseudo execute them respectively, and pseudo continue the placement of other pieces until all pieces have been placed into the strip to obtain a temporary height. Then we select a placement with the minimum temporary height among the N placements to execute.

We evaluate the performance of DHA on four groups of instances in the literature, which cover the small data sets and the large data sets. The computational results show that the DHA can achieve high quality solutions within reasonable times. And the comparisons with other state-of-the-art algorithms also show that the DHA is highly competitive.

The remaining part of the paper is organized as follows. Section 2 gives the specific description of the two-dimensional strip packing problem. In Section 3, some important concepts are defined, then the ingredients of our algorithm are described, including the constructive phase and the search phase. Sections 4 is dedicated to the computational results and concluding remarks are given in Section 5.

2. Problem specification

Let S be a rectangular strip with fixed width W and infinite height, and consider S embedded into a two-dimensional Cartesian reference frame such that the left-bottom corner coincides with the origin. Suppose there are n rectangular pieces going to be placed into the strip and each piece i has width w_i and height h_i. The objective is to place all n pieces without overlapping into the strip to minimize the total height of the pieces.

For each piece i placed into the strip, let (x_{i1},y_{i1}) and (x_{i2},y_{i2}) denote the coordinates of the left-bottom and right-top vertices, respectively. Then the problem can be formulated as follows:

$$min \max_{i=1}^{n}(y_{i2})$$

s.t.

1. $(x_{i2} - x_{i1}, y_{i2} - y_{i1}) \in \{(w_i,h_i),(h_i,w_i)\}$
2. $max(x_{i1} - x_{j2}, x_{j1} - x_{i2}, y_{i1} - y_{j2}, y_{j1} - y_{i2}) \geqslant 0$
3. $0 \leqslant x_{ik} \leqslant W,\ y_{ik} \geqslant 0,\ k \in \{1,2\}$

In constraints 1 to 3, i, j applies to $1,2,\ldots,n$ and $i \neq j$. Constraint 1 implies that each piece should be placed orthogonally in the strip. Constraint 2 prevents any two pieces from overlapping. And constraint 3 requires that all pieces should be placed completely in the strip. The placements should satisfy the three constraints and the objective is to place all the n pieces in the strip such that the total height of the pieces is minimized.

3. The deterministic heuristic algorithm

The deterministic heuristic algorithm (DHA) contains two phases: the rapid constructive phase and the partial tree search phase. During the constructive phase, the DHA places the rectangular pieces into the strip with a greedy strategy whose overall objective is to make the height of all placed pieces as low as possible. In addition, the DHA always makes the placing piece as close as possible with other pieces already placed and tries to make less adverse impact to the future placements. During the partial tree search phase, the DHA applies the depth-first search to find a most promising placement. Through pseudo executing the constructive phase of pieces and comparing the final total heights of different placements respectively, the DHA chooses a best placement according to the evaluation criterions and backtracks to execute the current placement.

We first define several important concepts used by the DHA, and describe the general framework of the DHA. Then we mainly introduce the constructive phase in detail and make a brief introduction about the partial tree search phase.

3.1. Definition

Definition 1 (*Configuration*). Suppose at the current moment, there are several pieces already in the strip while others remain to be placed, this is called a configuration. We present a configuration by the list of rectangular pieces and the empty space of strip. If there is no piece in the strip, we call it an initial configuration. If all pieces have been placed into the strip, we call it a final configuration.

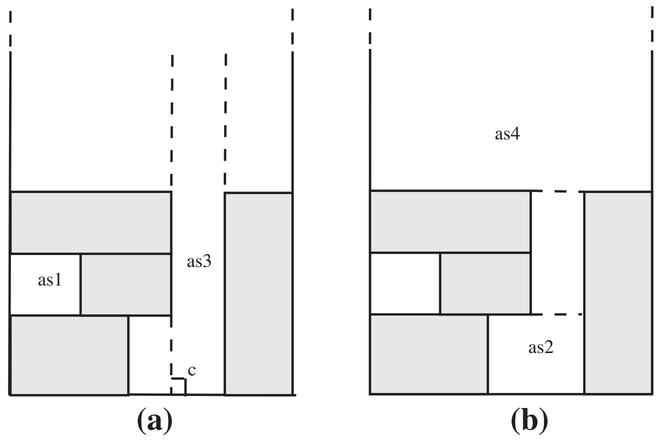

Fig. 1. All action spaces at the current configuration.

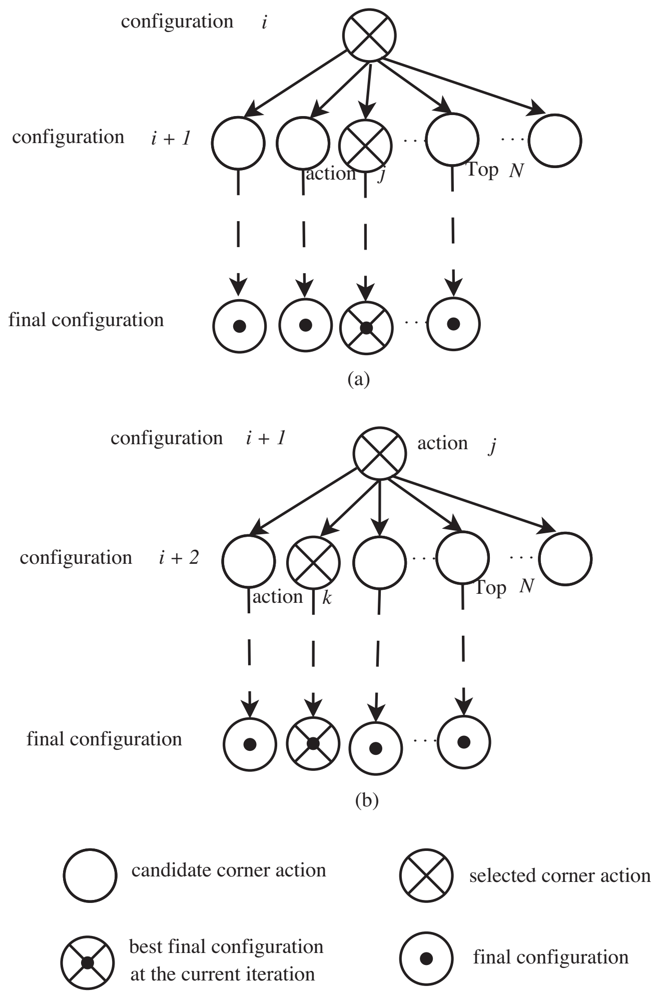

Fig. 2. The partial tree search phase.

Definition 2 (*Action space*). At the current configuration, we can view the empty space in the strip as many small rectangular strips. For these small strips, each one is large enough such that the left, right and bottom edges paste at least one piece or one edge of the outside big strip respectively, the top is open or be closed by a bottom edge of one piece. These small strips are called action spaces.

Fig. 1 gives an example of all action spaces at the current configuration. There are four action spaces: as1, as3 in Fig. 1(a) and as2, as4 in Fig. 1(b). The definition of action space is adapted and extended from the action space of the BFA algorithm proposed by He, Huang, and Jin (2012) and the FDA algorithm proposed by He and Huang (2011). Then we improved the constructive algorithm basing on the following two ideas. One of the advantages is that it is not necessary to place the pieces from the bottom empty space of strip to the top space. We usually search all action spaces to find the best placement to execute, such that the bottom empty space could be occupied at a suitable time when some outside pieces fit it best. The other is that we will consider more corners while using action spaces, including the virtual corners, as corner *c* in Fig. 1(a) shown, so it expands the search spaces such that it may improve the solution quality.

Definition 3 (*Corner action*). At the current configuration, there are only two corners in an action space, and the corner directions are "⌊" and "⌋". A corner action is an action that places a piece into a corner of an action space so that one vertex coincides with a corner.

Then we define a new criteria to evaluate different corner actions.

Definition 4 (*Compact degree*). The compact degree relates to four indicators, as shown in Eq. (1).

$$C_i = \langle e^{-p_i}, h_i, n_i, f_i \rangle \tag{1}$$

1. Paste number p_i denotes how many edges are pasted by the edges of the action space ($2 \leqslant p_i \leqslant 4$). The e^{-p_i} is a normalization of p_i. The larger the value of p_i the smaller the value of e^{-p_i}, then the placing rectangular piece is closer with the placed pieces in the strip.
2. Symbol h_i denotes the altitude of the action space from the bottom of strip. The h_i is smaller, then the height of the placing piece is lower.
3. Symbol n_i denotes the number of action spaces after the piece is placed in the strip. The n_i is smaller, then the empty space of strip is more smooth and it is easier for the next piece to be placed.
4. Symbol f_i denotes the influence of this current placement to the future placements. The f_i is related to a threshold value v_i. First, set v_i beforehand according to edges length of all pieces by sorting these edge lengths in non-ascending order and choosing a length at a position of 30% in the sorting queue as the value of v_i. Then after a piece is placed in a corner, we choose one new action space whose bottom edge is equal with the piece's, and then investigate the absolute difference between the width of the new action space and this threshold value. We consider the smaller one in priority because it is still possible for pieces to fill up this action space such that this space may not be wasted or a very small space will be wasted.. Hence, the f_i is smaller, the placement will influence less to the future placements.

Two compact degrees can be compared in the lexicographical order. The smaller the compact degree is, the better the placement is.

3.2. Algorithm description

In this subsection, we introduce the DHA in detail for solving the strip packing problem. The DHA applies the rapid constructive phase and the partial tree search phase jointly to ensure a high performance. The constructive phase plays an important role in

Table 1
Settings of important parameters.

Parameters	Value	Description
k%	50%	The selective percentage of action spaces
N%	50%	The selective percentage of corner actions
Lowerbound	50	The lowerbound of the corner actions
Upperbound	90	The upperbound of the corner actions

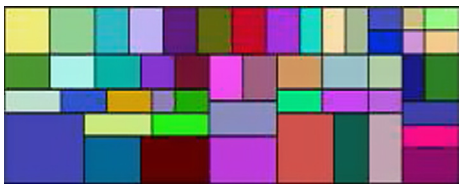

Fig. 3. The packing layout of benchmark P_1.

the DHA, which influences the solution quality and the computational time of the DHA. The detailed constructive phase is presented in Algorithm 1.

Algorithm 1. Pseudo-code of the constructive phase

```
1: Input: width $W$ of rectangular strip, $n$ pieces and best known height $h_{bk}$ of the strip
2: Output: the smallest height $h$ of strip achieved and the corresponding solution
3: $h \leftarrow 0$
4: Initialize the action space list with one action space that coincides with the strip
5: $H \leftarrow$ the total length of larger edge of pieces
6: All edges are sorted in non-ascending order and calculate threshold $v_i$
7: while There are still some pieces outside the strip do
8:    Sort the action spaces in non-descending order of the coordinate $y$ of their left-bottom vertices
9:    Pseudo place each remaining item into the top $k$ action spaces by different orientations to find all corner actions
10:    Calculate the compact degree of each corner actions
11:    Select a corner action in lexicographic order: < compact degree↓, area of pieces↑, height of pieces↑, coordinate $x$ of left_bottom vertices↓, prior to the horizontal placement >
12: repeat
13:       Update the action spaces
14:       if An action space is embedded by the placed piece then
15:          Perform the replacement of the action space
16:       end if
17:       if An action space is contained by other larger action spaces then
18:          Delete this action space
19:       end if
20:    until the list of action spaces is traversed
21:    $h \leftarrow$ the coordinate $y$ of the placed piece's right-top vertex
22: end while
23: return $h$ and the corresponding solution
```

During the constructive phase, we view the empty spaces of the strip as a union of all action spaces, and choose the best placement which includes the most suitable action space and piece. Furthermore, we define a new sorting rule for all candidate corner actions, which tries to make the pieces as compact as possible and tries to keep the total height of the pieces as low as possible. This reflects that the constructive phase is a greedy packing procedure.

The partial tree search phase selects the top N candidate placements which have smaller compact degrees, pseudo executes the constructive phase after each of the N placements is pseudo executed to get N final configurations, then compares the total heights of the pieces in the N final configurations, and then backtracks and selects one among the N candidate placements to execute. After all pieces have been placed in the strip, the obtained height of strip is just the smallest height among searched solutions. The detailed search procedure is shown in Fig. 2.

In Fig. 2(a), after a corner action is selected in configuration i, the top N corner actions are pseudo executed respectively, and then continue the constructive phase to final configurations. Then at configuration $i+1$, a corner action (action j) is selected according to the comparative result of the final configurations. Then the search procedure continues for configuration $i+1$ to $i+2$, as shown in Fig. 2(b). One observes that the search phase selects a corner action for each configuration, and repeats this selection until it attains to a final configuration.

4. Computational results

The algorithm was implemented in C programming language and run as a single process/thread. All tests were performed on a personal computer with 3.0 GHz CPU and 2.0 GB memory with a time limit of 1000 s. Table 1 gives the descriptions and settings of the parameters used in the DHA. We will first introduce the experimental benchmarks in Section 4.1, and then give the detailed computational results in Section 4.2.

Table 2
Computational results on C21.

Instance	n	W	h_{opt}	Burke's metaheuristic			FH		DHA	
				$BF+TS(h)$	$BF+SA(h)$	$BF+GA(h)$	h	$t(s)$	h	$t(s)$
C11	16	20	20	**20**	**20**	**20**	**20**	0.00	**20**	0.00
C12	17	20	20	21	**20**	21	**20**	0.01	21	0.30
C13	16	20	20	**20**	**20**	**20**	21	0.00	**20**	0.02
C21	25	40	15	16	16	16	16	0.02	**15**	0.59
C22	25	40	15	16	16	16	**15**	0.00	**15**	0.23
C23	25	40	15	16	16	16	**15**	0.00	**15**	0.06
C31	28	60	30	31	31	31	31	0.02	31	3.27
C32	29	60	30	32	31	32	31	0.02	32	5.14
C33	28	60	30	31	31	31	32	0.02	**30**	0.25
C41	49	60	60	62	61	62	61	0.16	61	20.86
C42	49	60	60	62	61	62	61	0.11	61	16.95
C43	49	60	60	61	61	62	61	0.08	**60**	14.62
C51	73	60	90	92	91	92	91	0.28	**90**	2.02
C52	73	60	90	92	91	92	**90**	0.00	**90**	15.62
C53	73	60	90	92	92	92	91	0.30	**90**	39.18
C61	97	80	120	122	122	122	121	0.83	121	169.53
C62	97	80	120	121	121	121	121	0.89	121	139.25
C63	97	80	120	122	122	122	121	0.76	**120**	2.70
C71	196	120	240	245	244	245	241	13.91	241	1000.00
C72	197	120	240	244	244	244	241	11.64	241	1000.00
C73	196	120	240	245	245	245	241	15.00	241	1000.00
optima#				2	3	2	5		11	

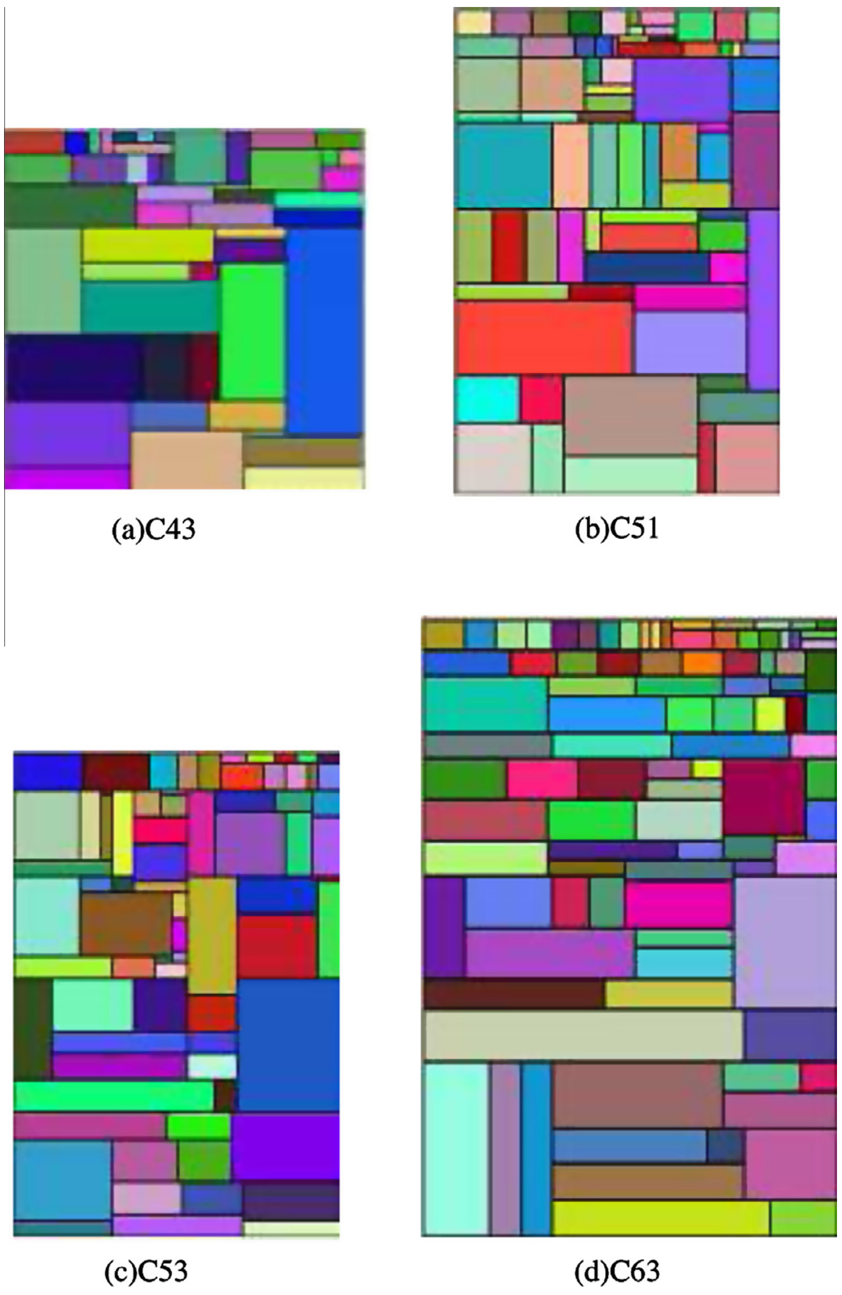

Fig. 4. The optimal layouts on four difficult instances of C21.

Table 3
Computational results on N13.

Instance	*n*	*W*	h_{opt}	Burke's metaheuristic			FH		DHA	
				BF + TS(*h*)	*BF + SA*(*h*)	*BF + GA*(*h*)	*h*	*t*(*s*)	*h*	*t*(*s*)
N1	10	40	40	**40**	**40**	**40**	**40**	0.00	**40**	0.02
N2	20	30	50	**50**	**50**	**50**	52	0.00	51	0.42
N3	30	30	50	51	51	52	51	0.02	**50**	0.20
N4	40	80	80	83	82	83	83	0.03	81	9.11
N5	50	100	100	103	103	104	102	0.08	103	13.91
N6	60	50	100	102	102	102	101	0.03	**100**	2.36
N7	70	80	100	105	104	104	102	0.13	102	41.50
N8	80	100	80	82	82	82	81	0.23	**80**	29.69
N9	100	50	150	152	152	152	151	0.14	**150**	0.25
N10	200	70	150	152	152	152	151	0.55	**150**	0.16
N11	300	70	150	153	153	153	151	1.48	**150**	0.61
N12	500	100	300	306	306	306	301	5.63	301	1000.00
N13	3152	640	960	964	964	964	**960**	1.91	**960**	191.67
optima#				2	2	2	2		8	

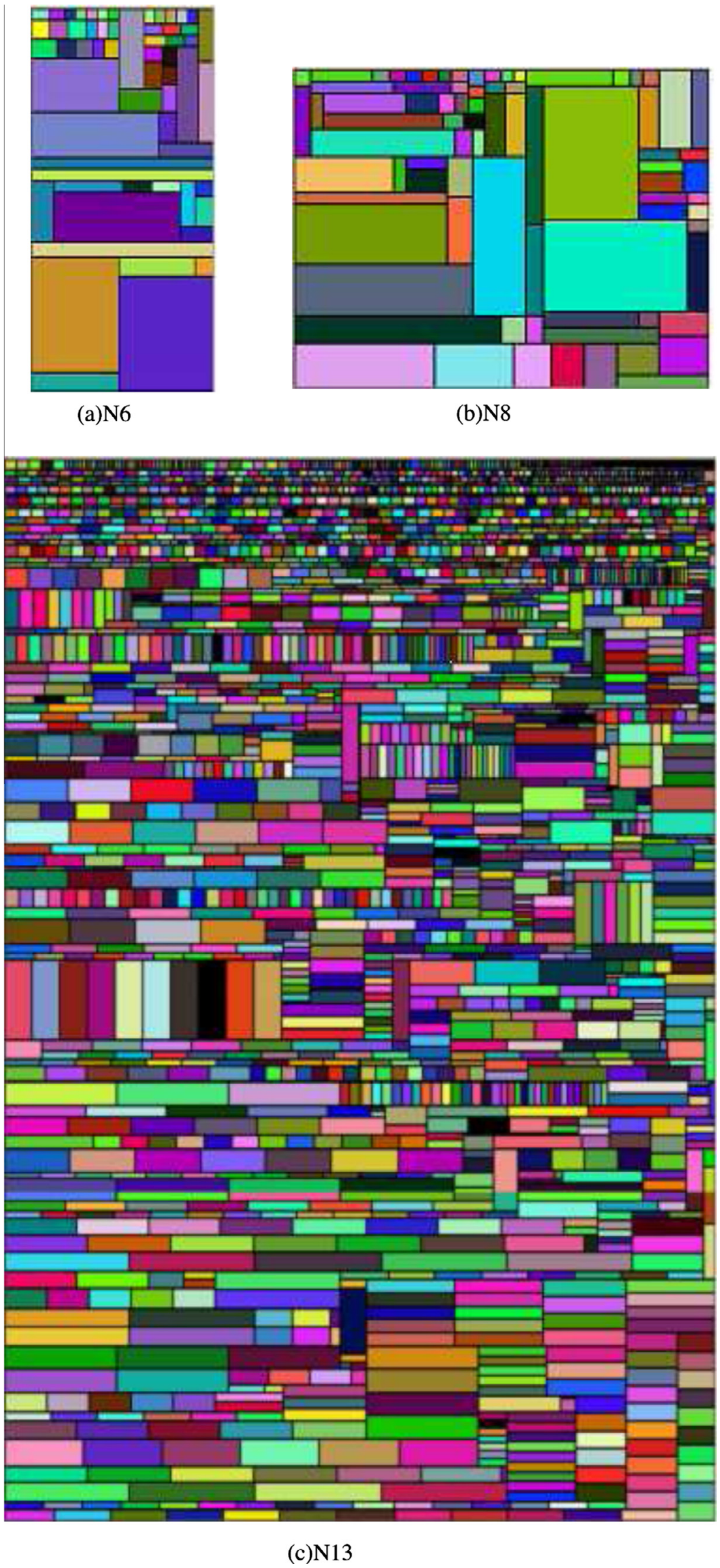

Fig. 5. The optimal layouts of three difficult instances of N13.

4.1. Test benchmarks

To evaluate the efficiency of the proposed DHA, the experiments were carried out on four well-known groups of benchmarks that are frequently used to assess the algorithms for solving 2D-SPP.

We call the four groups of benchmarks "C21", "N13", "CX" and "Beng" respectively.

- C21: 21 instances (C11-C73) refer to Hopper and Turton (2001)
- N13: 13 instances (N1-N13) refer to Burke et al. (2004)
- CX: 7 instances (50cx-15000cx) refer to Pinto and Oliveira (2005)
- Beng: 10 instances (beng1-beng10) refer to Bengtsson (1982)

4.2. Detailed computational results

In order to evaluate the efficiency of the DHA, we first introduce an example, and then compare the results obtained by the state-of-the-art algorithms in literature with the results obtained by the DHA on the four representative groups of benchmark instances.

We take a single benchmark "P_1" proposed by Ramesh Babu and Ramesh Babu (1999) as an example. There are 50 rectangular pieces need to be placed into a rectangular strip whose width $W = 1000$ and height is infinite. Experimental result shows that the DHA achieves a height of 375 in 0.16 s, and there is no wasted area in the strip, so it is an optimum solution. The detailed layout appears in Fig. 3.

4.2.1. Computational results on the commonly tested benchmark C21

For the benchmark *C*21, there are 21 small-scale instances which range from 16 to 197 rectangular pieces. These instances are classified by the number of rectangular pieces and are divided into seven categories with three ones each. The optimal solution of each instance is known in respect that each instance is constructed and there is no wasted area of strip. The benchmark *C*21 has been commonly tested by many researchers and the detailed information please refer to Hopper and Turton (2001).

Columns 1–4 in Table 2 present the features of the instances. Columns 5–11 present the computational results achieved by BF + TS, BF + SA, BF + GA (Burke et al., 2006), FH (Leung & Zhang, 2011) and DHA respectively, and the optimal solutions appear in bold. Algorithms BF + TS, BF + SA, BF + GA were run on a Pentium IV at 2.0 GHz and 10 times with a time limit of 60 s per run. Algorithm FH is a deterministic one, which was run on a 2 GHz Pentium 4 notebook with 2048 MB RAM with the time limit of 60 s per run.

Frome Table 2, one observes that BF + SA achieved three optimal solutions, BF + TS and BF + GA achieved two optimal ones respectively. The FH algorithm achieved five optimal solutions. And our DHA, which is also a deterministic algorithm, achieved 11 optimal solutions. Furthermore, the optimal solutions of the C21, C33, C43, C51, C53 and C63 instances were not achieved by BF + TS, BF + SA, BF + GA and FH algorithm.

Fig. 4 shows the optimal layouts achieved by the DHA on four difficult instances C43, C51, C53 and C63.

4.2.2. Computational results on the randomly generated benchmark N13

For benchmark N13, there are 13 instances which range from 10 to 3152 rectangular pieces. The N11, N12 and N13 instances belong to the large-scale data sets and the other ones belong to the small-scale data sets. These instances are randomly generated by Burke et al. and the optimal solution of each instance is known because the pieces are randomly cut continuously from a large rectangular piece. Besides, there are many small pieces and few large pieces for each instance. For the detailed information on benchmark N13, please refer to Burke et al. (2004).

Table 3 reports the detailed computational results obtained by BF + TS, BF + SA, BF + GA, FH and DHA, and the optimal solutions appear in bold. It can be observed that BF + TS, BF + SA, BF + GA and FH obtained two optimal solutions respectively. And the DHA achieved eight optimal solutions. Furthermore, the solutions of all instances achieved by the DHA are better than or equal to that of BF + TS, BF + SA, BF + GA and FH algorithms, except that the results on N2 and N5 a litter bit weaker. Besides, the DHA achieves two optimal solutions for three large-scale instances, which shows that the DHA is also effective for solving the instances with large data sets.

Fig. 5 shows the optimal layouts achieved by the DHA on three difficult instances N6, N8 and N13, and the optimal solutions of N6 and N8 have not been achieved by the BF + TS, BF + SA, BF + GA and FH algorithms.

4.2.3. Computational results on the extra large-scale benchmark CX

For the benchmark *CX*, there are seven instances which range from 50 to 15,000 rectangular pieces. These instances are all extra large-scale data sets except the 50cx and 100cx instances such that many algorithms could not handle so large data sets to get satisfied solutions within reasonable time. The optimal solutions of these instances are also known beforehand. For the detailed information on benchmark *CX*, please refer to Pinto and Oliveira (2005).

Table 4 shows the computational results of bottom-left solution, FH and DHA, and the optimal solutions appear in bold. The bottom-left solution did not achieve the optimal solutions, the FH achieved five optimal ones from seven instances. And the DHA had found four optimums and one near-optimal instance which is a litter bit worse than the FH. But the DHA is still good at solving the instances with large data sets. One analyses from the features of these large-scale instances that many pieces are of the same size, which shows these instances are weak heterogeneous, hence it is very efficient to be solved by the DHA in reasonable times even though some instances have a large number of pieces.

Table 4
Computational results on CX.

Instance	n	W	h_{opt}	Bottom-left	FH		DHA	
				h	h	$t(s)$	h	$t(s)$
50cx	50	400	600	674	624	0.33	644	26.91
100cx	100	400	600	679	619	3.56	637	266.64
500cx	500	400	600	692	**600**	2.00	601	1000.00
1000cx	1000	400	600	690	**600**	0.02	**600**	628.48
5000cx	5000	400	600	687	**600**	0.05	**600**	3.92
10000cx	10000	400	600	681	**600**	0.05	**600**	6.50
15000cx	15000	400	600	660	**600**	0.06	**600**	9.88
optima#				0	5		4	

Table 5
Computational results on Beng.

Instance	n	W	h_{lb}	Staircase		DHA	
				h	$t(s)$	h	$t(s)$
beng1	20	25	30	**30**	0.08	**30**	0.78
beng2	40	25	57	**57**	0.12	**57**	6.28
beng3	60	25	84	**84**	0.10	**84**	17.06
beng4	80	25	107	**107**	0.08	**107**	34.74
beng5	100	25	134	**134**	0.08	**134**	52.34
beng6	40	40	36	**36**	0.10	**36**	7.08
beng7	80	40	67	**67**	0.11	**67**	36.17
beng8	120	40	101	**101**	0.17	**101**	85.78
beng9	160	40	126	**126**	0.23	**126**	145.11
beng10	200	40	156	**156**	0.81	**156**	231.27
optima#				10		10	

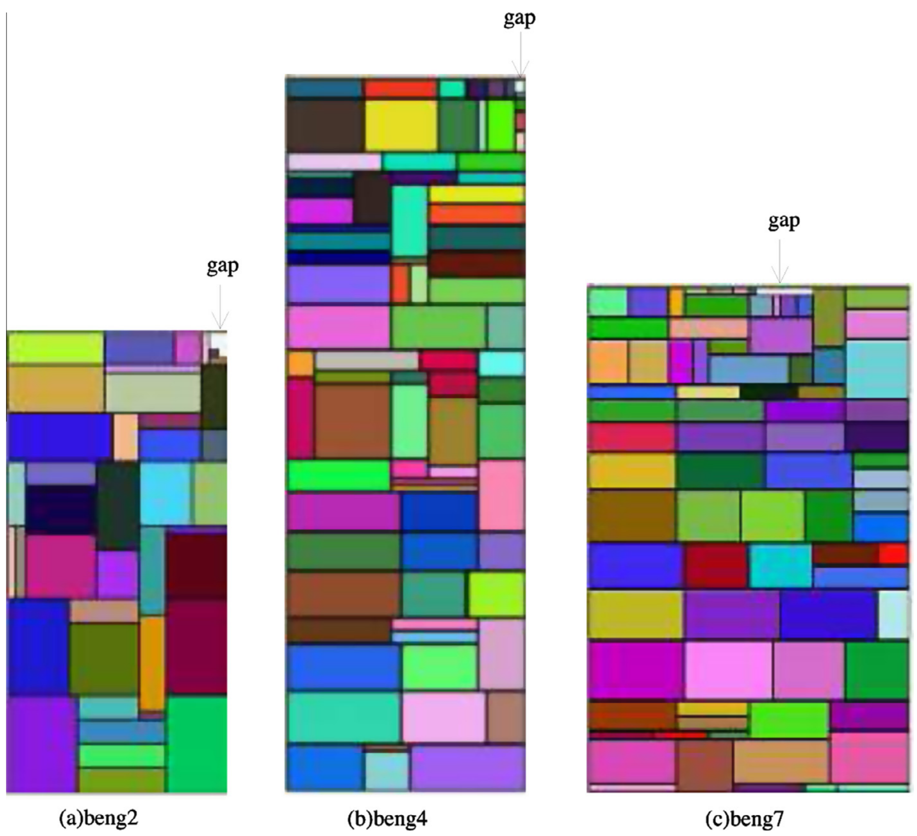

Fig. 6. The layouts of three instances of Beng.

4.2.4. Computational results on the general non-zero-waste benchmark Beng

For the benchmark *Beng*, there are ten small-scale instances which range from 20 to 200 rectangular pieces. The lower bounds (h_{lb}) of the strips are known but the optimal solutions are still not known. There are some wasted areas of the rectangular strip with lower bounds such that these instances are more general for the strip packing. For the detailed information on the benchmark *Beng*, please refer to Bengtsson (1982).

Table 5 presents the computational results achieved by the exact staircase algorithm and the deterministic heuristic algorithm DHA, and the optimal solutions appear in bold. The staircase algorithm was run on a Pentium 4 at 3.0 GHz and 1.0 GB memory with the time limit of 3600 s. From Table 5, we could clearly see that the results obtained by the two algorithms are equivalent, and they are the same as the lower bounds. Thus, the results show that DHA is still suitable for the general strip benchmark instances.

Fig. 6 shows the layouts achieved by the DHA on three instances Beng2, Beng4 and Beng7. The figures show that there still exists an empty gap of the strip by all pieces of each instance, which is different from other benchmarks mentioned above.

5. Conclusion

In this paper, we adapted the definition of action space on rectangular packing problem for the strip packing problem and proposed a deterministic heuristic algorithm (DHA) for the two-dimensional strip packing problem where 90° rotations are allowed. We mainly focused on how to define a good sorting rule to evaluate different placements so as to design an efficient constructive heuristic algorithm, which affects the performance of the tree search algorithm directly. The computational results on four well known groups of benchmarks showed that the proposed DHA are efficient on both small data sets and extra large data sets, and the DHA is also suitable for both zero-wasted benchmarks and non-zero-wasted benchmarks. In the future, we will further improve the performance of the proposed algorithm and extend it to solve problems without rotation constraints.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (Grant No. 61173180). Experiments were supported by the High Performance Computing Center experimental testbed in SCTS/CGCL.

References

Alvarez-Valdes, R., Parreño, F., & Tamarit, J. M. (2008). Reactive GRASP for the strip-packing problem. *Computers and Operations Research, 35*(4), 1065–1083.
Baker, B. S., Coffman, E. G., Jr., & Rivest, R. L. (1980). Orthogonal packings in two dimensions. *SIAM Journal on Computing, 9*(4), 846–855.
Belov, G., Scheithauer, G., & Mukhacheva, E. A. (2008). One-dimensional heuristics adapted for two-dimensional rectangular strip packing. *Journal of the Operational Research Society, 59*, 823–832.
Bengtsson, B. E. (1982). Packing rectangular pieces – a heuristic approach. *The Computer Journal, 25*, 353–357.
Burke, E. K., Kendall, G., & Whitwell, G. (2006). Metaheuristic enhancements of the best-fit heuristic for the orthogonal stock cutting problem. Computer science technical report no. NOTTCS-TR-SUB-0605091028-4370, University of Nottingham.
Burke, E. K., Hyde, M. R., & Kendall, G. (2011). A squeaky wheel optimization methodology for two-dimensional strip packing. *Computers and Operations Research, 38*(7), 1035–1044.
Burke, E. K., Kendall, G., & Whitwell, G. (2004). A new placement heuristic for the orthogonal stock-cutting problem. *Operations Research, 52*(4), 655–671.
Chazelle, B. (1983). The bottom-left bin packing heuristic: An efficient implementation. *IEEE Transaction on Computers, 32*(8), 697–707.

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

Christofides, N., & Whitlock, C. (1977). An algorithm for two-dimensional cutting problems. *Operation Research, 25*(1), 30–44.

Cui, Y., Yang, Y., Cheng, X., & Song, P. (2008). A recursive branch-and-bound algorithm for the rectangular guillotine strip packing problem. *Computers and Operations Research, 35*(4), 1281–1291.

He, K., & Huang, W. Q. (2011). An efficient placement heuristic for three-dimensional rectangular packing. *Computers and Operations Research, 38*(1), 227–233.

He, K., Huang, W. Q., & Jin, Y. (2012). An efficient deterministic heuristic for two-dimensional rectangular packing. *Computers and Operations Research, 39*(7), 1355–1363.

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

Huang, W. Q., & Chen, D. B. (2007). An efficient heuristic algorithm for rectangle-packing problem. *Simulation Modelling Practice and Theory, 15*(10), 1356–1365.

Kenmochi, M., Imamichi, T., Nonobe, K., Yagiura, M., & Nagamochi, H. (2009). Exact algorithms for the two-dimensional strip packing problem with and without rotations. *European Journal of Operational Research, 198*(1), 73–83.

Lesh, N., Marks, A., McMahon, A., & Mitzenmacher, M. (2004). Exhaustive approaches to 2D rectangular perfect packings. *Information Processing Letters, 90*(1), 7–14.

Leung, S. C. H., & Zhang, D. F. (2011). Fast layer-based heuristic for non-guillotine strip packing. *Expert Systems with Applications, 38*, 13032–13042.

Martello, S., Monaci, M., & Vigo, D. (2003). An exact approach to the strip packing problem. *INFORMS Journal on Computing, 15*(3), 310–319.

Pinto, E., & Oliveira, J. F. (2005). Algorithm based on graphs for the non-guillotinable two-dimensional packing problem. In *Second ESICUP Meeting*, Southampton.

Ramesh Babu, A., & Ramesh Babu, N. (1999). Effective nesting of rectangular parts in multiple rectangular sheets using genetic and heuristic algorithms. *International Journal of Production Research, 37*, 1625–1643.

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

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