

A Block Corner-Occupying Heuristic Algorithm for Constrained Two-Dimensional Guillotine Cutting Problem of Rectangular Items

WEIPING PAN 

College of Information Science and Engineering, Northeastern University, Shenyang 110819, China

e-mail: weiping209@126.com

This work was supported in part by the National Natural Science Foundation of China under Grant 61363026 and Grant 71371058.

ABSTRACT Aiming at the constrained two-dimensional guillotine cutting problem of rectangular items, a heuristic algorithm with block corner-occupying pattern is presented in this paper. It can maximize the pattern value for the totally included items, but the occurring frequency of each item type doesn't exceed its upper bound. Several rows and columns of identical items are packed at the left-bottom corner of the sheet, and the remaining part is divided into two sub-sheets. The sub-sheets are then packed and divided in the same way till no items can be packed. This upper bound and normal size methods applied in the algorithm will avoid the unnecessary calculation. The algorithm is compared with 9 literature algorithms with benchmark instances and random instances. Computational results show that, compared with the 8 heuristic algorithms, the pattern value of this algorithm is increased by 0.787% to 6.119% and the calculation time is reasonable. Compared with the exact algorithm, for large size instances the pattern value of this algorithm is 0.090% lower than it, but the calculation time is only 0.079% of it.

INDEX TERMS Corner-occupying pattern, constrained two-dimensional guillotine cutting problem, heuristic algorithm.

I. INTRODUCTION

In industrial production, for instance, the cutting of sheet metal, glass, and plywood, two-dimensional cutting problem often occurs. Good cutting pattern can improve the sheet utilization and reduce production cost [1]–[5]. Two-dimensional cutting problem can be regular or irregular cutting problem, constrained or unconstrained cutting problem, guillotine or non-guillotine cutting problem, depending on the item geometry, constraint on upper bound frequency of item, and cutting process, respectively.


This paper discuss the *rectangular constrained guillotine two-dimensional cutting*(RCG_2DC) problem: m types of rectangular items are cut from a rectangular sheet with size $L \times W$ (length L , width W) using guillotine cuts, where the i th type has size $l_i \times w_i$, value v_i , and upper bound b_i ($i \in M = \{1, 2, \dots, m\}$). The objective of RCG_2DC problem is to maximize the pattern value, which is the total value of items packed on the sheet. Assume that pattern P contains p_i pieces of type- i items, the pattern value of P is V . Let \mathbb{N} be the set of non-negative integers. The mathematical model of

the RCG_2DC problem is:

$$\begin{aligned} \max V &= \sum_{i=1}^m p_i v_i, \\ s.t. & (0 \leq p_i \leq b_i) \cap (p_i \in \mathbb{N}), \quad i = 1, \dots, m. \end{aligned} \quad (1)$$

Belonging to the NP difficult combinatorial optimization problem, RCG_2DC is called as a single large object packing problem (SLOPP) in [6], and the solution space of feasible cutting patterns is very large. The exact algorithms can only solve the small scale problems, and take time too long to solve the large scale problems, which is unbearable [7]–[10]. In practical cases, heuristic algorithms are generally used to solve the RCG_2DC. They can be divided into two types in terms of the construction idea.

The first type is the intelligent optimization algorithm. Alvarez et al. first proposed a greedy random adaptive search algorithm and developed a more complex tabu search algorithm, then implemented a path relinking procedure to improve the results of the above algorithm [11]. Hifi proposed a hybrid approach based on techniques of hill-climbing and dynamic programming [12]. Morabito and Pureza developed a heuristic algorithm based on graph search and dynamic programming [13].

The associate editor coordinating the review of this manuscript and approving it for publication was Ming Luo .

The second type is to limit the cutting pattern with a certain geometric feature to reduce the solution space, and thus simplifies the computation. For the two-staged cutting pattern, Lodi and Monaci built two integer linear programming models and used a branch-and-bound framework to test them [14]. Hifi and M'Hallah proposed an exact algorithm which based on a bottom-up strategy [15] and an approximation algorithm which based on beam search [16]. For the homogeneous T-shape cutting pattern, Cui proposed a tree-search algorithm which based on bottom-up approach [17]. Cui and Yang proposed a recursive branch-and-bound algorithm which based on top-down approach [18]. In order to improve the speed of the algorithm, Cui proposed a fast heuristic algorithm which based on dynamic programming and branch-and-bound techniques [19]. For the item corner-occupying cutting pattern, Chen proposed a recursive algorithm [20]. For the general T-shape cutting pattern, Cui and Huang proposed a heuristic algorithm with a layout-generation procedure [21]. For the homogenous strip corner-occupying cutting pattern, Cui and Chen proposed a recursion approach to consider a set of cutting patterns with specified geometric features, and used a bound technique to discard unpromising branches [22]. For the three-staged cutting pattern, Cui et al. proposed a heuristic algorithm with one exact procedure and two heuristic procedures [23].

Some of the above algorithms have shorter computation time but lower solution quality, while others have better solution quality but longer computation time. It is worth studying to construct an algorithm to get a good solution in a reasonable time.

In this paper, we propose a block corner-occupying heuristic algorithm for the RCG_2DC. The algorithm selects an item type, places several rows and columns of items at the left-bottom corner of the sheet and divides the rest of the sheet into two sub-sheets. The sub-sheets are further packed and divided until no items can be packed. There are three contributions in this paper. First, a new guillotine cutting pattern, namely block corner-occupying cutting pattern, is designed. Second, a heuristic algorithm based on dynamic programming for the block corner-occupying cutting pattern is constructed. Last, the proposed algorithm was compared with several published algorithms; the results show that the proposed algorithm is competitive.

The remainder of this paper is organized as follows. In section II we describe the characteristics and mathematical model of the block corner-occupying cutting pattern. In section III, we present a heuristic algorithm that generates the block corner-occupying cutting pattern. In section IV, we give the experimental results and compare the algorithm in this paper with those in literature. Finally, we conclude this work and make suggestions for future research in section V.

II. BLOCK CORNER-OCCUPYING CUTTING PATTERN

Without loss of generality, the length of the sheet and items are defined as the horizontal direction, and the width as the vertical direction. The items are allowed to rotate 90 degrees

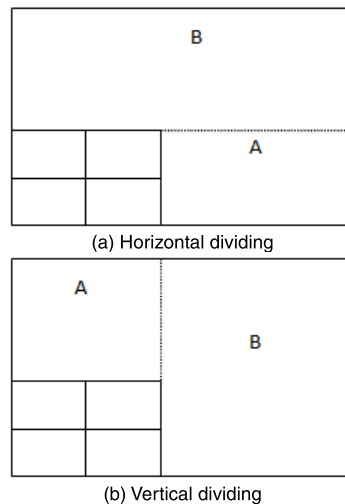


FIGURE 1. Two ways of divides the unoccupied region of the sheet.

in the packing process. After rotation, type- i item is converted into type- $(i+m)$ item, the length and width of the type- $(i+m)$ item is w_i and l_i , respectively, where $i \in M$. After this treatment, the original m types of items are converted into $2m$ new types. The type- i item and type- $(i+m)$ item correspond to the original and rotational case, respectively. In the following, if there is no special description, the type- i item refers to the new type- i item.

Definition 1 (Block): A block consists of several rows and columns of items with the same type and the same direction.

Definition 2 (Block Corner-Occupying Cutting Pattern): A type items are selected and packd at the left-bottom corner of the sheet in a block mode. A cutting line is drawn to divide the remaining part of the sheet into 2 sub-sheets. The sub-sheets are further packed and divided, until no item can be packed.

Definition 3 (Pattern Value): The pattern value is the total value of items contained in a sheet or a sub-sheet.

As shown in Figure 1, s rows and t columns of type- i items are placed at the left-bottom corner of the sheet $x \times y$, the unoccupied region is divided into sub-sheet A and B by a cutting line along the upper boundary (Figure 1a) or right boundary (Figure 1b). Let $f_X(x, y, i, s, t)$ and $f_Y(x, y, i, s, t)$ denotes the value of the sheet with the cutting line in horizontal and vertical direction, respectively. Let $F(x, y)$ be the value of the sheet and $n(x, y, i)$ be the number of type- i item in the sheet. Let $h(x, y, i)$ be the sum of the number of type- i items and type- $(i+m)$ items in the sheet $x \times y$.

When $1 \leq i \leq m$ and the cutting line is horizontal, (2) and (3) can be derived, as shown at the top of the next page.

When $1 \leq i \leq m$ and the cutting line is vertical, (4) and (5) can be derived, as shown at the top of the next page.

Equation (3) is explained as: s rows and t columns type- i items are placed at the left-bottom corner of the sheet $x \times y$, the unoccupied region is divided according to figure 1(a). If the total number of the type- i and type- $(i+m)$ items doesn't exceed the upper bound b_i , the value of sheet $x \times y$ is the sum

$$h(x, y, i) = n(x - tl_i, sw_i, i) + n(x, y - sw_i, i) + n(x - tl_i, sw_i, i + m) + n(x, y - sw_i, i + m) + st \quad (2)$$

$$f_X(x, y, i, s, t) = \begin{cases} v_i st + F(x - tl_i, sw_i) + F(x, y - sw_i) & \text{if } h(x, y, i) \leq b_i \\ 0 & \text{if } h(x, y, i) > b_i \end{cases} \quad (3)$$

$$h(x, y, i) = n(tl_i, y - sw_i, i) + n(x - tl_i, y, i) + n(tl_i, y - sw_i, i + m) + n(x - tl_i, y, i + m) + st \quad (4)$$

$$f_Y(x, y, i, s, t) = \begin{cases} v_i st + F(tl_i, y - sw_i) + F(x - tl_i, y) & \text{if } h(x, y, i) \leq b_i \\ 0 & \text{if } h(x, y, i) > b_i \end{cases} \quad (5)$$

of type-*i* items, sub-sheet A and B; otherwise it is 0. Equation (5) is similar to (3) except that the unoccupied region is divided according to figure 1(b).

When $m < i \leq 2m$ and the cutting line is horizontal, (6) and (7) can be derived, as shown at the top of the next page.

When $m < i \leq 2m$ and the cutting line is vertical, (8) and (9) can be derived, as shown at the top of the next page.

The descriptions of Equation (7) and Equation (9) are the same as that of Equation (3). $2m$ types items that can be placed at the left-bottom corner of sheet $x \times y$. Limited by the boundary of the sheet, the maximum number of rows and columns of type-*i* items at the left-bottom corner is $\lfloor y/w_i \rfloor$ and $\lfloor x/l_i \rfloor$, respectively. The symbol “[.]” represents the down rounding. $F_X(x, y)$ and $F_Y(x, y)$ represents the sheet value with the dividing line in horizontal and vertical direction, respectively, and are written as in (10)–(12), as shown at the top of the next page.

Equation (10) and (11) indicate that the optimal item type that placed at the left-bottom corner of the sheet and the optimal number of rows and columns are determined by maximizing the sheet value. Equation (12) indicates that the pattern value of the sheet is the larger one among two division modes.

III. HEURISTIC ALGORITHM

In this paper, all possible size of sub-sheet generated in optimal corner-occupying pattern is listed from small size to large. The corner-occupying pattern of the sheet is determined when the corner-occupying pattern of the sub-sheet $L \times W$ is obtained. The normal size and the pattern value upper bound of the sub-sheet are used to exclude unnecessary calculations.

A. NORMAL SIZE

The normal size is used to reduce the calculation in Chen (2008). In this paper, the normal length/width is a linear combination of the length/width of all items. Let G_L and G_W be the set of normal length and width, respectively, then:

$$G_L = \left\{ x | x = \sum_{i=1}^{2m} a_i l_i, \quad a_i \in N \text{ and } 0 \leq x \leq L \right\} \quad (13)$$

$$G_W = \{ y | y = \sum_{i=1}^{2m} b_i w_i, \quad b_i \in N \text{ and } 0 \leq y \leq W \} \quad (14)$$

In Equation (13-14), N is a set of non-negative integers. The normal size of the sub-sheet has the following properties: if the maximum normal length $A_L(x)$ and width $A_W(y)$ are not greater than x and y , the sub-sheet $A_L(x) \times A_W(y)$ and $x \times y$ have the same pattern value in accordance with the block corner-occupying cutting pattern.

It can be seen from the nature of the normal size that the algorithm of this paper only needs to investigate the normal size of the sub-sheet. Let $G_L = \{g_L(1), g_L(2), \dots, g_L(|G_L|)\}$, $G_W = \{g_W(1), g_W(2), \dots, g_W(|G_W|)\}$, where $|G_L|$ and $|G_W|$ are the number of elements of the set of G_L and G_W , respectively. The elements in the sets are arranged in ascending order.

For example when $l_1 = 45, l_2 = 68, l_3 = 83, l_4 = 91, L = 150$, then $G_L = \{45, 68, 83, 90, 91, 113, 128, 135, 136\}$, $|G_L| = 10$. Obviously, $|G_L|$ is much smaller than L .

Since $|G_L|$ and $|G_W|$ are much smaller than L and W , the application of normal size can avoid the unnecessary calculation and reduce the calculation time.

B. THE UPPER BOUND OF PATTERN VALUE

For the cutting problem with the same sheet and items, the total value of the items contained in the unconstrained pattern is greater than or equal to that in the constrained pattern. The reason is the number of times allowed for each item type is unconstrained in the unconstrained pattern, while it cannot exceed its upper bound in the constraint pattern. The unconstrained cutting problem is generally easier to solve. In this paper, the value of the optimal unconstrained block corner-occupying pattern is used as the upper bound of the constrained corner-occupying pattern. Let $f_X^U(x, y, i, s, t)$ and $f_Y^U(x, y, i, s, t)$ be the value of unconstrained pattern of the two dividing method in Figure 1, respectively. $F_X^U(x, y)$, $F_Y^U(x, y)$ represents the unconstrained pattern value of sheet with the dividing line in horizontal and vertical direction, respectively. $F^U(x, y)$ be the value of unconstrained pattern of sub-sheet $x \times y$. There are (15)–(19), as shown at the top of the next page.

The above equations constitute a dynamic programming model. Equations (15) and (16) are recursive formulas for the model. Equation (15) shows that s rows and t columns type-*i* items are placed at the left-bottom corner of the sheet $x \times y$, the unoccupied region is divided according to

$$h(x, y, i - m) = n(x - tl_i, sw_i, i) + n(x, y - sw_i, i) + n(x - tl_i, sw_i, i - m) + n(x, y - sw_i, i - m) + st \quad (6)$$

$$f_X(x, y, i, s, t) = \begin{cases} v_i st + F(x - tl_i, sw_i) + F(x, y - sw_i) & \text{if } h(x, y, i - m) \leq b_{i-m} \\ 0 & \text{if } h(x, y, i - m) > b_{i-m} \end{cases} \quad (7)$$

$$h(x, y, i - m) = n(tl_i, y - sw_i, i) + n(x - tl_i, y, i) + n(tl_i, y - sw_i, i - m) + n(x - tl_i, y, i - m) + st \quad (8)$$

$$f_Y(x, y, i, s, t) = \begin{cases} v_i st + F(tl_i, y - sw_i) + F(x - tl_i, y) & \text{if } h(x, y, i - m) \leq b_{i-m} \\ 0 & \text{if } h(x, y, i - m) > b_{i-m} \end{cases} \quad (9)$$

$$F_X(x, y) = \max_{i \in M} \max_{s \in \{1, \dots, \lfloor y/w_i \rfloor\}, t \in \{1, \dots, \lfloor x/l_i \rfloor\}} f_X(x, y, i, s, t) \quad (10)$$

$$F_Y(x, y) = \max_{i \in M} \max_{s \in \{1, \dots, \lfloor y/w_i \rfloor\}, t \in \{1, \dots, \lfloor x/l_i \rfloor\}} f_Y(x, y, i, s, t) \quad (11)$$

$$F(x, y) = \max\{F_X(x, y), F_Y(x, y)\} \quad (12)$$

$$f_X^U(x, y, i, s, t) = v_i st + F^U(x - tl_i, sw_i) + F^U(x, y - sw_i) \quad (15)$$

$$f_Y^U(x, y, i, s, t) = v_i st + F^U(tl_i, y - sw_i) + F^U(x - tl_i, y) \quad (16)$$

$$F_X^U(x, y) = \max_{i \in M} \max_{s \in \{1, \dots, \lfloor y/w_i \rfloor\}, t \in \{1, \dots, \lfloor x/l_i \rfloor\}} f_X^U(x, y, i, s, t) \quad (17)$$

$$F_Y^U(x, y) = \max_{i \in M} \max_{s \in \{1, \dots, \lfloor y/w_i \rfloor\}, t \in \{1, \dots, \lfloor x/l_i \rfloor\}} f_Y^U(x, y, i, s, t) \quad (18)$$

$$F^U(x, y) = \max\{F_X^U(x, y), F_Y^U(x, y)\} \quad (19)$$

figure 1(a). The value of sheet $x \times y$ is the sum of type- i items, sub-sheet A and B. The meaning of equation (16) is similar to that of equation (15), except that the unoccupied region is divided according to figure 1(b). Equations (17) and (18) show that the type, rows and columns of items are determined according to the principle of maximum pattern value. Equation (19) indicates that the pattern value of the sheet equals the larger of the pattern value of figure 1(a) and figure 1(b).

Due to the full capacity of the dynamic programming algorithm, the pattern values of sub-sheet with other sizes are known when that of the sub-sheet with largest size is obtained.

C. CALCULATION OF PATTERN VALUE

In the process of generating constrained pattern, the number of each item type in the pattern is kept to be within its upper bound. For Figure 1, s rows and t columns of type- i items are placed at the left-bottom corner of the sub-sheet $x \times y$, where $i \in \{1, \dots, 2m\}$. The number of type- j items contained in the sub-sheet A and B are $n(x_A, y_A, j)$ and $n(x_B, y_B, j)$, respectively. x_A and y_A are the length and width of the sub-sheet A, while x_B and y_B are the length and width of the sub-sheet B. For any feasible pattern, the number of each type items with original case and rotation case cannot be more than their upper bound. That is to say the following 4 constraints

must be satisfied. The heuristic algorithm in this paper is shown in Figure 2.

Constraint 1:

When $(1 \leq j \leq m)$ and $(j = i \text{ or } i - m)$, (20) can be obtained, as shown at the bottom of the next page.

Constraint 2:

When $(m + 1 \leq j \leq 2m)$ and $(j = i \text{ or } i + m)$, (21) can be obtained, as shown at the bottom of the next page.

Constraint 3:

When $(1 \leq j \leq m)$ and $(j \neq i \text{ and } i - m)$, (22) can be obtained, as shown at the bottom of the next page.

Constraint 4:

When $(m + 1 \leq j \leq 2m)$ and $(j \neq i \text{ and } i + m)$ and (23) can be obtained, as shown at the bottom of the next page.

Formula (20) means when type- j item is the original case of type- i item, the number of original case and rotational case of type- j items in pattern is st plus the number of type- j items and type- $(j + m)$ items in sub-sheet A and B. It is not more than the upper bound of type- j item.

Formula (21) means when type- j item is the rotation case of type- i item, the number of original case and rotational case of type- $(j - m)$ item is st plus the number of type- j items and type- $(j - m)$ items in sub-sheet A and B. It is not more than the upper bound of type- $(j - m)$ item.

Formula (22) means when type- j item is the original case and neither the type- i item nor the type- $(i - m)$ item, the

```

Input: an instance of RCG_2DC problem  $I^\# = (L, W, l[ ], w[ ], v[ ], b[ ])$ .
Output: the maximum pattern value of the instance  $I^\#$ .
0 The original item type- $i$  is regarded as two new item types of item- $i$  and item- $(i+m)$ , where  $i \in M$ .
1 Let  $F(x, y) = 0$ , for  $x = 0, \dots, L, y = 0, \dots, W$ .
2 For  $p = 1$  to  $|G_L|$ 
3   For  $q = 1$  to  $|G_W|$ 
4     Let  $x = g_L(p), y = g_W(q)$ 
5     If  $F^U(x, y) \leq F(x, g_W(q-1))$  then  $q = q + 1$ 
6     For  $i = 1$  to  $2m$ 
7       If  $x < l_i$  or  $y < w_i$  then the item type- $i$  is not considered
8       For  $s = 1$  to  $\min\{y/w_i, b_i\}$ 
9         For  $t = 1$  to  $\min\{x/l_i, b_i/s\}$ 
10          If  $F(x, y) < v_i st + F(x - tl_i, sw_i) + F(x, y - sw_i)$  and the pattern is feasible
11            Let  $F(x, y) = v_i st + F(x - tl_i, sw_i) + F(x, y - sw_i)$ 
12          If  $F(x, y) < v_i st + F(tl_i, y - sw_i) + F(x - tl_i, y)$  and the pattern is feasible.
13            Let  $F(x, y) = v_i st + F(tl_i, y - sw_i) + F(x - tl_i, y)$ 
14 For  $x = 1$  to  $L$ 
15   For  $y = 1$  to  $W$ 
16     Let  $F(x, y) = F(A_L(x), A_W(y))$ .
    
```

FIGURE 2. A heuristic algorithm for solving the pattern value of the sub-sheet.

TABLE 1. Hardware environment of the algorithms in this paper and nine published papers.

Algorithm	Hardware environment
This paper	PC Main frequency 2.4GHz, Memory 4MB
[15]	UltraSparc10 processor, Main frequency 250 MHz, Memory 128 MB
[20]	PC Main frequency 300MHz, Memory 128MB
[21]	PC Main frequency 2.66GHz, Memory 3.37GB
[22]	PC Main frequency 2.66GHz, Memory 3.37GB
[16]	PC Main frequency 2.8GHz Memory 512 MB
[23]	PC Main frequency 2.13GHz, Memory 4GB
[10]	PC Main frequency 3.60GHz, Memory 16GB
[18]	PC Mian frequency 2.20GHz, Memory 1GB
[19]	PC Main frequency 2.66GHz, Memory 3.37GB

number of original case and rotational case of type- j item is the number of type- j items and type- $(j+m)$ items in sub-sheet A and B. It is not more than the upper bound of type- j item.

Formula (23) means when type- j item is the rotation case and neither the type- i item nor the type- $(i+m)$ item, the number of original case and rotational case of type- $(j-m)$

item is the number of type- j items and type- $(j-m)$ items in sub-sheet A and B. It is not more than the upper bound of type- $(j-m)$ item.

From the figure 2, we can see that the time complexity of this algorithm is less than $O(2m |G_L| |G_W| b_i)^2 + LW$. The 0th line indicates that the items have two cases with/without

$$st + n(x_A, y_A, j) + n(x_B, y_B, j) + n(x_A, y_A, j + m) + n(x_B, y_B, j + m) \leq b_j \tag{20}$$

$$st + n(x_A, y_A, j) + n(x_B, y_B, j) + n(x_A, y_A, j - m) + n(x_B, y_B, j - m) \leq b_{j-m} \tag{21}$$

$$n(x_A, y_A, j) + n(x_B, y_B, j) + n(x_A, y_A, j + m) + n(x_B, y_B, j + m) \leq b_j \tag{22}$$

$$n(x_A, y_A, j) + n(x_B, y_B, j) + n(x_A, y_A, j - m) + n(x_B, y_B, j - m) \leq b_{j-m} \tag{23}$$

TABLE 2. Experimental results of the small and middle size benchmark instances.

ID	[15]		[20]		[21]		[22]		[10]		This paper
	pattern value	gap	pattern value	gap	pattern value	gap	pattern value	gap	pattern value	gap	pattern value
HH	10689	11.545	-	-	11391	4.670	-	-	-	-	11923
2	2535	14.241	2892	0.138	2594	11.642	2892	0.138	-	-	2896
3	1720	10.465	1860	2.151	1740	9.195	1840	3.261	-	-	1900
A1	1820	14.286	2020	2.970	1820	14.286	1940	7.216	-	-	2080
A2	2315	9.287	2502	1.119	2315	9.287	2455	3.055	-	-	2530
STS2	4450	4.719	4620	0.866	4620	0.866	4620	0.866	-	-	4660
STS4	9409	3.486	9700	0.381	9700	0.381	9700	0.381	-	-	9737
CHL1	8360	6.651	8660	2.956	8474	5.216	8660	2.956	-	-	8916
CHL2	2235	7.114	2326	2.923	2273	5.323	2292	4.450	-	-	2394
CW1	6402	5.654	6402	5.654	6402	5.654	6402	5.654	6766	-0.030	6764
CW2	5354	6.108	5354	6.108	5354	6.108	5354	6.108	5689	-0.141	5681
CW3	5287	8.644	5689	0.967	5468	5.048	5689	0.967	5744	0.000	5744
Hch12	9630	3.873	9954	0.492	9735	2.753	9891	1.132	-	-	10003
Hch19	5100	2.353	5240	-0.382	5060	3.162	5220	0.000	-	-	5220
2s	2430	14.650	2778	0.288	2604	6.989	2778	0.288	-	-	2786
3s	2599	6.618	2721	1.838	2623	5.642	2721	1.838	-	-	2771
A1s	2950	1.186	2950	1.186	2950	1.186	2950	1.186	-	-	2985
A2s	3423	4.557	3535	1.245	3451	3.709	3535	1.245	-	-	3579
STS2s	4569	2.320	4653	0.473	4653	0.473	4653	0.473	-	-	4675
STS4s	9481	3.196	9770	0.143	9642	1.473	9770	0.143	-	-	9784
OF1	2713	0.885	2737	0.000	2713	0.885	2737	0.000	2757	-0.725	2737
OF2	2515	10.099	2690	2.937	2690	2.937	2690	2.937	2769	0.000	2769
W	2623	5.642	2721	1.838	2721	1.838	2721	1.838	-	-	2771
CHL1s	13036	1.258	13099	0.771	13036	1.258	13099	0.771	-	-	13200
CHL2s	3162	5.787	3279	2.013	3236	3.368	3266	2.419	-	-	3345
A3	5380	3.420	5451	2.073	5403	2.980	5451	2.073	-	-	5564
A4	5885	6.117	6179	1.068	6014	3.841	6179	1.068	-	-	6245
A5	12553	4.987	12976	1.564	12779	3.130	12985	1.494	-	-	13179
CHL5	363	10.193	390	2.564	363	10.193	390	2.564	-	-	400
CHL6	16572	1.979	16869	0.184	16573	1.973	16869	0.184	-	-	16900
CHL7	16728	1.028	16840	0.356	16695	1.228	16881	0.113	-	-	16900
CU1	12321	1.453	12330	1.379	12321	1.453	12330	1.379	12500	0.000	12500
CU2	26100	0.383	26100	0.383	25934	1.026	26100	0.383	26200	0.000	26200
Hch13s	11961	3.771	12214	1.621	12059	2.927	12214	1.621	-	-	12412
Hch14s	11408	7.574	12002	2.250	11964	2.574	11964	2.574	-	-	12272
Hch16s	60170	2.187	61040	0.731	60666	1.352	61040	0.731	-	-	61486
Hch17s	62459	1.479	63102	0.445	62845	0.856	63102	0.445	-	-	63383
Hch18s	729	23.320	904	-0.553	825	8.970	876	2.626	-	-	899
Average		6.119		1.398		4.101		1.752		-0.128	

Note: "-" indicates the data is not available.

rotation, the item type- i is converted into new item type- $(i + m)$ is after rotation. The 1th line indicates that the pattern values of sub-sheet of all the possible sizes were initialized. Lines 2-4 indicate that the normal size of the sheets was

investigated one by one from small to large. In the 5th line, the current sub-sheet will not be investigated if its value upper bound is not greater than that of the smaller sub-sheet. Lines 6-7 indicate that the item type of the left-bottom corner

TABLE 3. Experimental results of the large size benchmark instances.

ID	[16]		[21]		[22]		[23]		[10]		This paper
	pattern value	gap	pattern value	gap	pattern value	gap	pattern value	gap	pattern value	gap	pattern value
ATP30	140197	0.504	140461	0.315	140904	0.000	140544	0.256	140904	0.000	140904
ATP31	820260	0.570	822417	0.306	823976	0.116	822417	0.306	824878	0.007	824934
ATP33	37973	0.250	38015	0.139	38068	0.000	38017	0.134	38068	0.000	38068
ATP33	235580	0.562	235580	0.562	236611	0.123	236549	0.15	236903	0.000	236903
ATP34	357741	1.277	359162	0.876	361357	0.264	359806	0.696	361952	0.099	362310
ATP35	614429	1.317	618397	0.666	621021	0.241	618397	0.666	622518	0.000	622518
ATP36	129262	1.317	130156	0.622	130744	0.169	130366	0.459	130965	0.000	130965
ATP37	385811	0.461	385811	0.461	387276	0.081	386064	0.395	387439	0.038	387588
ATP38	259137	0.955	260622	0.380	261395	0.083	260622	0.38	261625	-0.005	261612
ATP39	266378	1.089	267684	0.595	268750	0.196	267772	0.562	269278	0.000	269278
ATP40	65584	2.607	66032	1.911	67154	0.208	66224	1.616	67294	0.000	67294
ATP41	202305	3.631	206190	1.679	206542	1.505	205159	2.19	210713	-0.504	209651
ATP42	33012	2.011	33289	1.163	33566	0.328	33289	1.163	33756	-0.237	33676
ATP43	212062	2.507	214589	1.300	214651	1.271	212558	2.268	218820	-0.659	217379
ATP44	70940	7.253	72895	4.376	73438	3.604	73048	4.158	76122	-0.049	76085
ATP45	74205	0.655	74205	0.655	74691	0.000	74205	0.655	74691	0.000	74691
ATP46	147021	2.695	149658	0.885	149911	0.715	149658	0.885	150983	0.000	150983
ATP47	144317	5.342	146789	3.568	148540	2.347	147811	2.852	152778	-0.492	152026
ATP48	165428	3.174	165640	3.042	167427	1.942	165640	3.042	170678	0.000	170678
ATP49	211784	4.941	213667	4.016	216749	2.537	213874	3.915	222248	0.000	222248
Average		2.156		1.376		0.787		1.337		-0.090	

of the sub-sheet has $2m$ selections. When the item size is larger than the sub-sheet size, the item is not considered. Lines 8-9 indicate that the numbers of row and column of the item are enumerated, the item will not exceed the sub-sheet boundary and the item number will not exceed its upper bound. Lines 10-11 indicate that the pattern value is larger if the sub-sheet is horizontally divided, the number of each item type is below its upper bound, and the sub-sheet value is renewed by the current pattern value. Lines 12-13 indicate the vertical division case. Lines 14-16 indicate that the pattern value of a sub-sheet with all possible sizes is equal to that of a sub-sheet with the normal size.

IV. COMPUTATIONAL EXPERIMENTS

This section presents computational results of the presented algorithm in section III. The algorithm is coded in C# and running on a personal computer. Two groups of benchmark instances and one group of random instances are used to compare this algorithm with nine algorithms in published papers. Table 1 lists the hardware environment of algorithms in this paper and the other published papers. The calculation time of the algorithm in this paper is obtained by record the running time of the algorithm in solving the instances. The calculation time of the published algorithm is obtained from the published papers.

A. BENCHMARK INSTANCES OF SMALL AND MIDDLE SIZE

This group consists of 38 benchmark instances with small and middle size, as used in [21]. Table 2 shows the statistical results of the pattern value in this paper and four other published papers. The “gap” represents the pattern value difference in term of percentage between this paper and others. $\text{Gap} = (V_{\text{this algorithm}} - V_{\text{published algorithm}}) / V_{\text{published algorithm}} \times 100$. Positive gap means higher pattern value in this paper. The pattern value of different algorithms is derived from table 2 in [15], section 3 in [20], table 3 in [21], section 3 in [22] and table 4 in [10].

The averaged calculation time for each instance of algorithm is 1.07 seconds in [15], 5.12 seconds in [20], 0.014 seconds in [21]; 0.83 seconds in [10], it is not reported in [22]. The total calculation time is 6.82 seconds for all instances and the averaged time is 0.18 seconds in this paper.

Compared to the algorithm in [15], the pattern values of the algorithm in this paper are all higher in 38 instances, the pattern value is averaged increased by 6.119%. Compared to the algorithm in [20], the number of instances with higher, equal, lower pattern value in this paper is 34, 1, 2, respectively, and the average increase of pattern value is 1.398%. Compared to the algorithm in [21], the pattern values of all 38 instances are higher in this paper, and the average increase of pattern value is 4.101%. Compared to the algorithm in [22], the number of

TABLE 4. Experimental results of the random instances.

ID	[18]		[19]		This paper
	pattern value	gap	pattern value	gap	pattern value
1A_1	1759161	4.698	1759161	4.698	1841806
1A_2	2676987	12.942	2660962	13.622	3023433
1A_3	2452608	1.327	2452608	1.327	2485161
1A_4	2036218	8.967	2036218	8.967	2218813
1A_5	2713122	4.637	2713122	4.637	2838925
1A_6	2906130	3.414	2906130	3.414	3005337
1A_7	2251501	2.347	2196233	4.923	2304354
1A_8	1752346	4.806	1752346	4.806	1836562
1A_9	2558842	7.160	2532439	8.277	2742051
1A_10	2570413	1.780	2541538	2.936	2616154
1A_11	1889186	5.548	1883293	5.879	1994006
1A_12	3386985	1.251	3386985	1.251	3429359
1A_13	2910872	3.047	2910872	3.047	2999566
1A_14	2213106	4.878	2213106	4.878	2321059
1A_15	1956502	6.811	1956502	6.811	2089767
1A_16	2570444	6.913	2570444	6.913	2748146
1A_17	2179374	6.075	2179374	6.075	2311781
1A_18	2916082	4.749	2916082	4.749	3054572
1A_19	2897755	5.760	2897755	5.760	3064662
1A_20	2572052	4.743	2572052	4.743	2694032
Average		5.093		5.386	

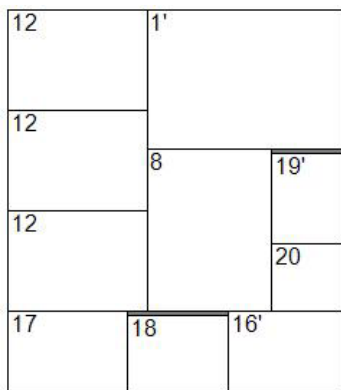


FIGURE 3. Solution to A3 (pattern value 5564).

instances with higher, equal pattern value in this paper is 35, 2, respectively, and the average increase of pattern value is 1.752%. Compared to the algorithm in [10], the number of instances with equal, lower pattern value in this paper is 4, 3, respectively, and the average decrease is 0.128%.

Figure 3 is the cutting pattern of instance A3 generated by the algorithm of this paper. The number in the figure represents the type of item. The symbol “/” on the upper right corner of the number indicates that the item has rotated 90 degrees. Gray area represents the scrap of the sheet.

B. BENCHMARK INSTANCES OF LARGE SIZE

This group consists of 20 benchmark instances with large size, as used in [16]. The first 10 instances are un-weighted where the item value is equal to the item area, and the

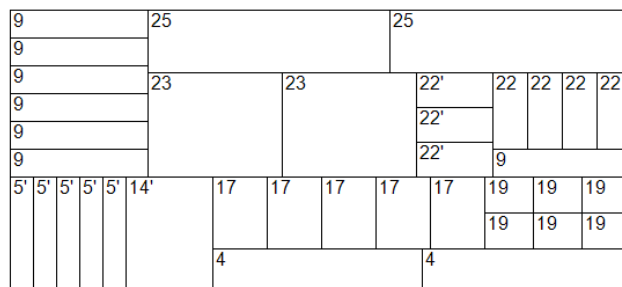


FIGURE 4. Solution to ATP36 (pattern value 130942).

last 10 are weighted where the item value may be not equal to the item area. Table 3 shows the statistical results of the pattern value in this paper and five other published papers. The pattern value of different algorithm is derived from table 1 in [16], table 4 in [21], section 3 in [22], table 2 in [23], and table 3 in [10].

The averaged calculation time for each instance of algorithm is 0.20 seconds in [16], 0.014 seconds in [21], 35.22 seconds in [22], 0.18 seconds in [23], 701.83 seconds in [10] and 0.56 seconds in this paper. It should be noted that the algorithm in [10] is exact algorithm and the maximum solution time was set to 900 seconds.

Compared to [16], the pattern value in this paper is higher in all 20 instances. The average increase is 2.156%. Compared to [21], the pattern value is higher in all 20 instances and the average increase is 1.376%. Compared to [22], the pattern value is higher in 17 instances, equal in 3 instances, and the average increase is 0.787%. Compared to [23], the

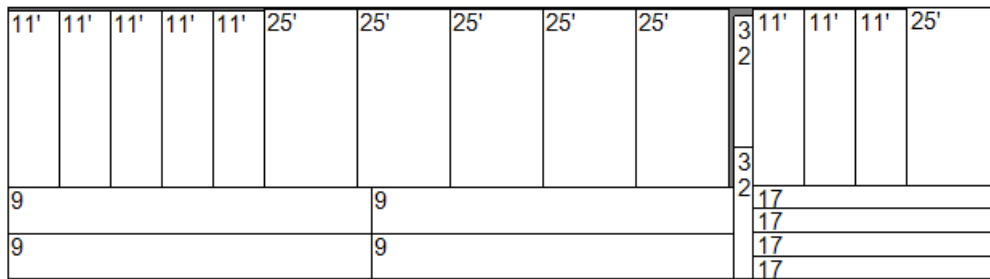


FIGURE 5. Solution to ATP48 (pattern value 170678).



FIGURE 6. Solution to 1A_14 (pattern value 2321059).

pattern value is higher in all 20 instances, and the average increase is 1.337%. Compared to [10], the pattern value in this paper is higher in 3 instances, equal in 11 instances, lower in 6 instances, the average decrease is 0.09%. Figure 4 and figure 5 are the cutting patterns of instance ATP36 and ATP48 generated by the algorithm of this paper, respectively. As show in figure 5 and 6 of [23], the pattern value of instance ATP 36 and ATP 48 is 130366 and 165640, respectively. They are all lower than the pattern value in this paper.

C. RANDOM INSTANCES WITH SIZE OF TYPICAL IN PRACTICE

This group consists of 20 random instances with size of typical in practice, as used in [18]. Table 5 shows the statistical results of the pattern value in this paper and two published papers. The averaged calculation time on each instance is 0.37 seconds in [18], 0.014 seconds in [19], and 1.69 seconds in this paper. Compared to [18], the pattern values in this paper are higher in all 20 instances, the average increase is 5.093%. Compared to [19], the pattern values in this paper are better in all 20 instances, the average increase is 5.386%. Figure 6 is the cutting pattern of instance 1A_14 generated by the algorithm of this paper.

V. CONCLUSION

With regarding to the RGC_2DC problem, a block corner-occupying cutting heuristic algorithm is presented, where rectangular item is rotated by 90 degrees for higher pattern value. This algorithm is a deterministic algorithm, that is

to say, the result of each run of the algorithm is the same. This type of pattern is a superset of item corner-occupying cutting pattern and strip corner-occupying cutting pattern, as far as we know, it has never appeared in the literature. This pattern can make the same type of rectangular items gathered in a block as most as possible and is beneficial for the sheet cutting process. The presented algorithm is a heuristic algorithm for guillotine cutting pattern. Compared with other heuristic algorithms, this algorithm has higher pattern value than two-staged, homogenous T-shape, item corner-occupying, homogenous strip corner-occupying, T-shape and three-staged cutting algorithm. This algorithm can keep calculation time at reasonable level to meet the requirement in practical application. The pattern value of the algorithm in this paper is very close to that of the exact algorithm, and the calculation time is much less than the exact algorithm.

The design idea in this paper is a relatively simple and convenient reference for the enterprise staff to develop the cutting software. Combine the presented algorithm with sequential value correction heuristic algorithm to solve the two-dimensional cutting stock problem of rectangular items can be the research in future.

REFERENCES

- [1] K. Kim, B. I. Kim, and H. Cho, "Multiple-choice knapsack-based heuristic algorithm for the two-stage two-dimensional cutting stock problem in the paper industry," *Int. J. Prod. Res.*, vol. 52, no. 19, pp. 5675–5689, 2014.
- [2] Y. Wang, R. Xiao, and H. Wang, "A flexible labour division approach to the polygon packing problem based on space allocation," *Int. J. Prod. Res.*, vol. 55, no. 11, pp. 3025–3045, 2017.
- [3] B. Guo, Z. Liang, Q. Peng, Y. Li, and F. Wu, "Irregular packing based on principal component analysis methodology," *IEEE Access*, vol. 6, pp. 62675–62686, 2018.
- [4] Z. Chen and J. Chen, "An effective corner increment-based algorithm for the two-dimensional strip packing problem," *IEEE Access*, vol. 6, pp. 72906–72924, 2018.
- [5] B. Guo, Y. Ji, J. Hu, F. Wu, and Q. Peng, "Efficient free-form contour packing based on code matching strategy," *IEEE Access*, vol. 7, pp. 57917–57926, 2019.
- [6] G. Wäscher, H. Haußner, and H. Schumann, "An improved typology of cutting and packing problems," *Eur. J. Oper. Res.* vol. 183, no. 3, pp. 1109–1130, Dec. 2007.
- [7] N. Christofides and C. Whitlock, "An algorithm for two-dimensional cutting problems," *Oper. Res.* vol. 25, no. 1, pp. 30–44, Jan./Feb. 1977.
- [8] M. Hifi, "An improvement of viswanathan and bagchi's exact algorithm for constrained two-dimensional cutting stock," *Comput. Oper. Res.* vol. 24, no. 8, pp. 727–736, Aug. 1997.
- [9] V.-D. Cung, M. Hifi, and B. L. Cun, "Constrained two-dimensional cutting stock problems a best-first branch-and-bound algorithm," *Int. Trans. Oper. Res.* vol. 7, no. 3, pp. 185–210, May 2000.

- [10] A. S. Velasco and E. Uchoa, "Improved state space relaxation for constrained two-dimensional guillotine cutting problems," *Eur. J. Oper. Res.*, vol. 272, no. 1, pp. 106–120, Jan. 2019.
- [11] R. Alvarez-Valdés, A. Parajón, and J. M. Tamarit, "A tabu search algorithm for large-scale guillotine (un)constrained two-dimensional cutting problems," *Comput. Oper. Res.*, vol. 29, no. 7, pp. 925–947, Jun. 2002.
- [12] M. Hifi, "Dynamic programming and hill-climbing techniques for constrained two-dimensional cutting stock problems," *J. Combinat. Optim.*, vol. 8, no. 1, pp. 65–84, Mar. 2004.
- [13] R. Morabito and V. Pureza, "A heuristic approach based on dynamic programming and and/or-graph search for the constrained two-dimensional guillotine cutting problem," *Ann. Oper. Res.* vol. 179, no. 1, pp. 297–315, Sep. 2010.
- [14] A. Lodi and M. Monaci, "Integer linear programming models for 2-staged two-dimensional Knapsack problems," *Math. Program.*, vol. 94, nos. 2–3, pp. 257–278, Jan. 2003.
- [15] M. Hifi and R. M'Hallah, "An exact algorithm for constrained two-dimensional two-staged cutting problems," *Oper. Res.* vol. 53, no. 1, pp. 140–150, 2005.
- [16] M. Hifi, R. M'Hallah, and T. Saadi, "Algorithms for the constrained two-staged two-dimensional cutting problem," *Inform. J. Comput.*, vol. 20, no. 2, pp. 212–221, 2008.
- [17] Y. Cui, "An exact algorithm for generating homogenous T-shape cutting patterns," *Comput. Oper. Res.* vol. 34, no. 4, pp. 1107–1120, Apr. 2007.
- [18] Y. Cui and Y. Yang, "A recursive branch-and-bound algorithm for constrained homogenous T-shape cutting patterns," *Math. Comput. Model.* vol. 54, nos. 5–6, pp. 1320–1333, Sep. 2011.
- [19] Y. Cui, "Fast heuristic for constrained homogenous T-shape cutting patterns," *Appl. Math. Model.* vol. 36, no. 8, pp. 3696–3711, Aug. 2012.
- [20] Y. Chen, "A recursive algorithm for constrained two-dimensional cutting problems," *Comput. Optim. Appl.* vol. 41, pp. 337–348, Dec. 2008.
- [21] Y. Cui and B. Huang, "Heuristic for constrained T-shape cutting patterns of rectangular pieces," *Comput. Oper. Res.* vol. 39, no. 12, pp. 3031–3039, Dec. 2012.
- [22] Y. Cui and Q. Chen, "Simple heuristic for the constrained two-dimensional cutting problem," *Proc. Inst. Mech. Eng., B, J. Eng. Manuf.*, vol. 226, no. 3, pp. 565–572, Mar. 2012.
- [23] Y. P. Cui, Y. Cui, T. Tang, and W. Hu, "Heuristic for constrained two-dimensional three-staged patterns," *J. Oper. Res. Soc.*, vol. 66, no. 4, pp. 647–656, Apr. 2015.
- [24] M. Hifi and C. Roucairol, "Approximate and exact algorithms for constrained (Un) weighted two-dimensional two-staged cutting stock problems," *J. Combinat. Optim.* vol. 5, no. 4, pp. 465–494, 2001.



WEIPING PAN received the B.E. degree in mathematics and applied mathematics from the Fujian Agricultural and Forestry University, Fuzhou, China, and the M.E. degree in computer system architecture from Guangxi University, Nanning, China, in 2011 and 2015, respectively. He is currently pursuing the Ph.D. degree in systems engineering with Northeast University, Shenyang, China. His research interests include optimal calculation, and modeling and optimization.

• • •