Computer Science and Information Systems 17(3):717–735 https://doi.org/10.2298/CSIS200125017Y

An Improved Heuristic-Dynamic Programming Algorithm for Rectangular Cutting Problem

Aihua Yin, Chong Chen, Dongping Hu, Jianghai Huang and Fan Yang

School of Software and Internet of Things Engineering, Jiangxi University of Finance and Economics, Nanchang Jiangxi, China.
Dongping_hu337@jxufe.edu.cn

Abstract. In this paper, the two-dimensional cutting problem with defects is discussed. The objective is to cut some rectangles in a given shape and direction without overlapping the defects from the rectangular plate and maximize some profit associated. An Improved Heuristic-Dynamic Program (IHDP) is presented to solve the problem. In this algorithm, the discrete set contains not only the solution of one-dimensional knapsack problem with small rectangular block width and height, but also the cutting positions of one unit outside four boundaries of each defect. In addition, the denormalization recursive method is used to further decompose the sub problem with defects. The algorithm computes thousands of typical instances. The computational experimental results show that IHDP obtains most of the optimal solution of these instances, and its computation time is less than that of the latest literature algorithms.

Keywords: Guillotine, Two-dimension cutting problem, Dynamic programming, Defect, NP-hard.

1. Introduction

The two-dimensional cutting problem with defects is a research hotspot of combinatorial optimization. In the industrial manufactural area, many 2D cutting problems will encounter defects. For example, in the furniture industry, the wood panels may contain damaged areas which cannot be used for furniture panel surfaces. In the steel industry, some coils may contain defects that cannot be used as construction materials. Natural products such as leather usually have cut marks, so the defective parts can hardly be used on the surface of goods. In the literatures, the existing algorithms [1-2] for defect free problems are relatively extensive, and the research on multiple defects and guillotine cut has attracted more and more attention in recent years.

Experts have proposed many algorithms on the two-dimensional cutting problem without defect. In the latest literature, Wang et al. (2017) [2] propose a heuristic search algorithm based on grouping rules, which designs the key complement of the large and small parts division strategy and the quick recommendation of the block. Song et al. (2010) [3] propose a heuristic algorithm based on dynamic programming, which uses a subset of all possible cutting pattern and is an incomplete algorithm. Wuttke and Heese(2017) [4] propose a sequential heuristic with feedback loop and formulate the

sequencing problem as a mixed integer program in the two-dimension cutting problem. They use real data to test their heuristic and illustrate its applicability to a problem of realistic size. Yoon et al. (2013) [5] propose an improved version of the cutting problem for solving standard two-dimensional cutting problem, and their algorithm removes the dominated patterns efficiently and avoids duplicated patterns. Herz (1972) [6] uses a discretization set of all necessary cutting positions to propose an accurate recursive process. Beasley (1985) [7] shows how to improve the performance of the recursive process Herz's discretization sets and introduces a heuristic correction of the algorithm which limits the number of cuts in the discretization sets.

Now there have been more and more literatures on the issue of the two-dimensional cutting problem with defects. Carnieri et al. (1993) [8] propose a heuristic dynamic programming algorithm including branch and bound search, but they only study the two-dimensional cutting problem with one defect. Vianna and Arenales (2006) [9] re-examine this problem by providing an AND/OR-based branch-qualification algorithm that further introduce a heuristic search that combines depth-first search and depth-limiting and hill-climbing strategies. Neidlein and Wäscher (2008) [10] reduce the size of discretization sets in the algorithm proposed by Vianna and Arenales (2006) [9], however, their algorithms do not obtain optimal solutions. Afsharian et al. (2014) [11] modify the predecessor's heuristic dynamic programming algorithm to solve the problem with 4 defects. Their discretization sets size are cumbersome, which means the computational efficiency is not high. Martin M. et al. (2019) [12] propose a compact integer linear programming (ILP) model for the problem based on the discretization of the defective object and develop a Benders decomposition algorithm and a constraint-programming (CP) based algorithm as solution methods. For the non-guillotine cutting problem, Gonçalves and Wäscher (2020) [13] combine a MIP model with a new hybrid algorithm to solve it and Birgin et al. (2020) [14] propose a mixed integer linear programing model for the problem with usable leftovers. Velasco and Eduardo (2019) [15] study the constrained two-dimensional guillotine cutting problem for obtaining upper bounds. Russo et al. (2020) [16] review the best exact and heuristic solutions for C2DC and reviewed and classified the available upper bound. Wu et al. (2019) [17] discuss the same problem but They don't publish the source of their data.

In this paper, inspired by the previous algorithms [7,11], an improved heuristic dynamic programming algorithm is proposed to solve the problem with multiple defects in the way of guillotine cut. The algorithm reduces the discretization sets size of Afsharian et al. (2014) [11]. However, the cut positions at one unit from the four boundaries of the defect are added to the new discrete set. The computational results show that the algorithm improves the computational efficiency on thousands of typical instances.

Section 2 of the paper presents a description of the problem. Section 3 gives a detailed description of the algorithm and prove two important theorems about the complexity of the algorithm. Section 4 gives the calculation results of thousands of typical examples, and compares the algorithm in this paper with the best algorithm at present. Section 5 draws the conclusion.

2. Problem Description

For the convenience of later description, table 1 shows a list of the symbols with their meanings to be used.

Table 1. The list of the symbols with their meanings

sym	meaning	sym	meaning
W_0	the width of the large object	H_0	the height of the large object
w_i^s	the width of i-th small rectangular block	h_i^s	the height of i-th small rectangular block
d_j	the j-th defect	p_i	the number of cuts of i-th small rectangular block
w_j^d	the width of j-th defect	h_j^d	the height of j-th defect
x_j^d	the x-axis of j-th defect	y_j^d	the y-axis of j-th defect
z_x	the vertical cutting position	z_y	the horizontal cutting position
$\Phi_x^c(x)$	the vertical discretization set of the C-block	$\Phi_y^c(y)$	the horizontal discretization set of the C-block
$\Phi_x^D(x)$	the vertical discretization set of the D-block	$\Phi_y^c(y)$	the horizontal discretization set of the D-block
v_i	the value of i-th small rectangular block		

Let m different types of small rectangular blocks $i(i = 1, 2, \dots, m)$, each associated with an integer width w_i^s, an integer height h_i^s and the profit value v_i, must be cut from a single rectangular large object with a width of W_0 and a height of H_0 to maximize the total value of the small rectangular blocks produced by the cutting process. The solution of the problem is a cutting pattern, a form of small rectangular blocks produced from large object and a description of the layout in which the small rectangular blocks are arranged on large object. In this layout, all small rectangular blocks must be arranged parallel to the large object which is called a feasible solution of the problem. To establish the Cartesian coordinate system, let the bottom-left vertex of the large object be at the origin, the x-axis and y-axis be coincident with the wide and high edges of the object respectively. So, the large object can be represented by $(0, 0, W_0, H_0)$. For the issues to be considered, the following constraints should be met:

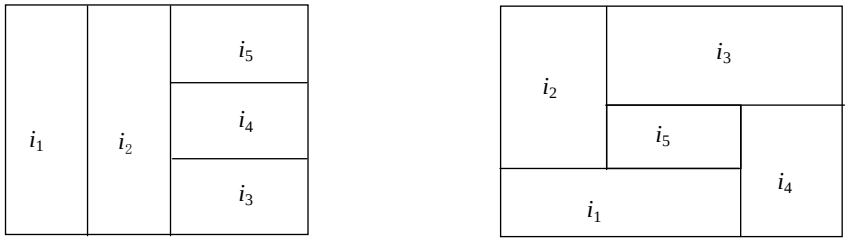
- The number of each type of small rectangular block cut is unlimited, that is $p_i \geq 0$;
- When cutting any small rectangular blocks, the given length and width orientation must be maintained, and 90 ° rotation is not allowed;
- Every cutting action must be guillotine mode, i.e., each cutting action exactly divides the current sheet into two parts (see Fig. 1);
- Every small rectangular block cut from the large object can not contain any defects, and its lower left coordinate shall be (x_i^s, y_i^s), that is $0 \leq x_i^s \leq W_0 - w_i^s, 0 \leq y_i^s \leq H_0 - h_i^s$, it must meet the requirements: $(x_j^d + w_j^d \leq x_i^s)\ or\ (x_j^d \geq x_i^s + w_i^s)\ or\ (y_j^d + h_j^d \leq y_i^s)\ or\ (y_j^d \geq y_i^s + h_i^s)$.

The cutting problem solved in this paper requires that all of the above constraints to be satisfied, that is, it is a two-dimensional, unconstrained, guillotine, single large object cutting patterns problem with defects (2D_UG_SLOPP_D). 2D_UG_SLOPP_D is generalization of 2D_UG_SLOPP.

Let P be a feasible solution of the problem and $p_i \geq 0$ $(i = 1, 2, \dots, m)$ be the amount of the i-th small rectangular block cut from the large object in P, then $(p_1, p_2, \dots, p_m)$ is use to describe the feasible cutting pattern in this paper. The goal of the problem is to

maximize the value of the small blocks cut from the large object, and the object function of the problem can be expressed as follows:

$$\begin{cases} \max\ V = \sum_{i=1}^{m} v_i * p_i \\ s.t.\ (p_1, p_2, \dots, p_m)\ \ is\ a\ feasible\ cutting\ pattern \end{cases} \quad (\mathbf{1})$$



(a) A guillotine cutting mode (b) A Non-guillotine cutting mode

Fig. 1. Two cutting modes: guillotine and non-guillotine

A defect is actually an irregular figure. Considering the cutting method here, it is appropriate to use rectangular area to express defects. Let n defects be in the large object, the j^{th} defect $d_j (j = 1, 2, \dots, n)$ have a width $w^d{}_j$ and a height $h^d{}_j$, and its bottom-left vertex on the large plate be (x_j^d, y_j^d), then d_j can be represented by $(x_j^d, y_j^d, w_j^d, h_j^d)$ (see Fig. 2).

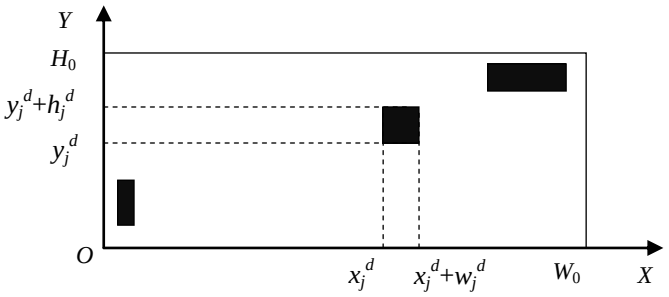


Fig. 2. The defects on the large object and their representation

3. Algorithm Description

The algorithm in this paper is called an improved heuristic dynamic programming algorithm (IHDP). It combines quasi human idea with dynamic programming algorithm. Using dynamic programming algorithm to solve 2D_UG_SLOPP_D, the resulting

subproblem is either 2D_UG_SLOPP_D or 2D_UG_SLOPP. Adopting different methods to solve these two different problems is the critical improvement of this algorithm.

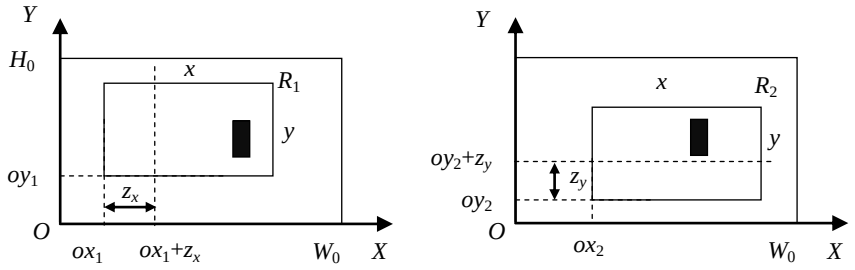
3.1. Basic definition

For the convenience of the following description, here are two important definitions.

Definition 1 (**sub-block**). In the guillotine mode, multiple rectangles which are neither the small rectangular blocks nor the wastes are formed after the large plate is cut several times. These rectangles are called sub-block. In this paper, the large object is regarded as the largest sub-block, and a sub-block corresponds to a sub-problem and vice versa.

Let the coordinates of a sub-block be (ox, oy), the width and the height of it be x and y respectively, then the sub-block is represented by R=(ox, oy, x, y).

According to this definition, two sub-blocks R_1=(ox_1, oy_1, x, y) and R_2=(ox_2, oy_2, x, y) with different coordinates in the sub-block are different sub-problems, even if they have the same size. For a vertical cut (parallel to the y-axis) on the sub-block R_1=(ox_1, oy_1, x, y) at the cut position z_x, two smaller sub-blocks are (ox_1, oy_1, z_x, y) and (ox_1+z_x, oy_1, x-z_x, y). Similarly, for a horizontal cut (parallel to the x-axis) at the cut position z_y on R_2, two sub-blocks (ox_2, oy_2, x, z_y) and (ox_2, oy_2+z_y, x, y-z_y) are also formed (see Fig. 3).



(a) a vertical cut on a sub-block (b) a horizontal cut on a sub-block

Fig. 3. The guillotine cut on the sub-block (vertical or horizontal) results in two smaller sub-blocks

Definition 2 (**C-block and D-block**). For a sub-block R=(ox, oy, x, y), if it contains a defect or overlap with any defect, it is called a D-block; otherwise it is called a C-block.

3.2. Discretization Sets

The cutting position set on the sub-block is a discretization set. If the sub-block is a C-block, the discretization sets [8] are defined by the equations (2), (3), and (4). Otherwise, the discretization sets are defined by the equations (5), (6), and (7). Z+ belongs to a positive integer set.

Discretization sets of C-blocks. If a sub-block is a C-block, the discretization sets defined by the equations (2), (3), and (4) are quite same with the discretization sets proposed by Afsharian et al. [11]. They are established by the solution of a one-dimensional knapsack problem with the width and height of the small rectangular blocks. Let $\boldsymbol{\Phi}_{\boldsymbol{x}}^{\boldsymbol{C}}(\boldsymbol{x})$ and $\boldsymbol{\Phi}_{\boldsymbol{y}}^{\boldsymbol{C}}(\boldsymbol{y})$ denote the vertical discretization set and the horizontal discretization set of sub-blocks respectively, they are described as follows:

$$\Phi_x^C(x) = \left\{ z_x | z_x = \sum_{i=1}^{m} \alpha_i w_i^s, 1 \le z_x \le x - w_0/2, \alpha_i \in Z^+ \cup \{0\}, \forall i \right\} \quad (\mathbf{2})$$

$$\Phi_y^C(y) = \left\{ z_y | z_y = \sum_{i=1}^{m} \beta_i h_i^s, 1 \le z_y \le y - h_0/2, \beta_i \in Z^+ \cup \{0\}, \forall i \right\} \quad (\mathbf{3})$$

$$w_0 = min\{w_i^s : h_i^s < y\} \ , h_0 = min\{h_i^s : w_i^s < x\}, i = 1, 2, \dots, m \quad (\mathbf{4})$$

Discretization sets of D-blocks. If a sub-block is a D-block, the discretization sets defined by the equations (5), (6), and (7). These discretization sets add the cutting position of one unit outside four boundaries of each defect into $\boldsymbol{\Phi}_{\boldsymbol{x}}^{\boldsymbol{C}}(\boldsymbol{x})$ and $\boldsymbol{\Phi}_{\boldsymbol{y}}^{\boldsymbol{C}}(\boldsymbol{y})$. They reduce the discretization sets proposed by Afsharian et al. [11].

$$\Phi_x^D(x) = \Phi_x^C(x) \cup \left\{ \begin{matrix} z_x | z_x = \delta_j \left(x_j^d - ox - 1 \right) or\ \gamma_j \left(x_j^d + w_j^d - ox + 1 \right), \\ \delta_j, \gamma_j \in \{0,1\}, \ j = 1, 2, \dots, n \end{matrix} \right\} \quad (\mathbf{5})$$

$$\Phi_y^D(y) = \Phi_y^C(y) \cup \left\{ \begin{matrix} z_y | z_y = \mu_j \left(y_j^d - oy - 1 \right) or\ \theta_j \left(y_j^d + h_j^d - oy + 1 \right), \\ \mu_j, \theta_j \in \{0,1\}, \ j = 1, 2, \dots, n \end{matrix} \right\} \quad (\mathbf{6})$$

$$w_0 = min\{w_i^s : h_i^s < y\} \ , h_0 = min\{h_i^s : w_i^s < x\}, i = 1, 2, \dots, m \quad (\mathbf{7})$$

3.3. Dynamic programming

This algorithm is an improved heuristic algorithm based on dynamic programming. For subproblems without defects (C-block), IHDP uses the method of Herz (1972) [6] and Beasley (1985) [7] to construct recursive function $F(x, y)$ for solving it. In this paper, the upper bound of discretization set is extended to $x - w_0/2$, $y - h_0/2$. For the sub-problem with defects (D-block), IHDP adopts a denormalization recursive function $F(ox, oy, x, y)$ which is different against Afsharian et al. [11] to deal with it. Furthermore, the cutting positions of one unit outside four boundaries of each defect are added into the discretization sets.

It is easy to get a lower bound of the objective function of C-blocks which can be got by the function $g(x, y)$. Every time, the sub-blocks are divided into the same type small rectangular blocks, and the lower bound is the maximum value of m cutting pattern. The functions $F(x, y)$ and $g(x, y)$ are as follows:

$$g(x,y) = max\left\{ v_i \cdot \left\lfloor \frac{x}{w_i^s} \right\rfloor \cdot \left\lfloor \frac{y}{h_i^s} \right\rfloor \right\}, w_i^s < x, h_i^s < y, i = 1,2, \dots, m \quad (\mathbf{8})$$

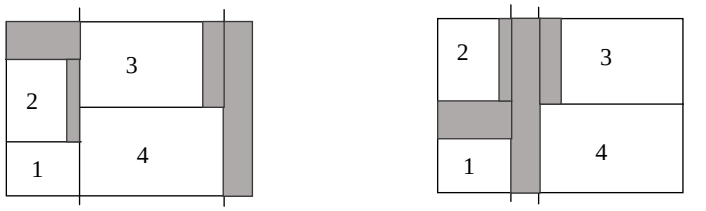
$$F(x,y) = max\begin{Bmatrix} g(x,y), \\ F(z_x,y) + F(p(x-z_x),y),\ z_x \in \Phi_x^C(x), 1 \le z_x \le \frac{x}{2}, \\ F(x,z_y) + F\left(x, q(y-z_y)\right), z_y \in \Phi_y^C(y), 1 \le z_y \le \frac{y}{2}, \\ x \in \Phi_x^C(W_0) \cup \{W_0\},\ \ y \in \Phi_y^C(H_0) \cup \{H_0\} \end{Bmatrix} \quad (\mathbf{9})$$

Where, if $x < w_0$ *or* $y < h_0$, then $F(x, y)$=0. Due to the appearance of the repeated cutting pattern, the discretization set of vertical (or horizontal) in (9) is limited to half the width (height) of the sub-block. In addition, Beasley (1985) [7] has proved that a kind of normalized cutting pattern will not result in the optimization of the solution of the recursive equation to solve the C-block. This pattern is to arrange the waste at the bottom left of the C-block (see Fig. 4), which is implemented with the two functions $p(x)$ and $q(y)$ introduced in the above recursive function. These two functions are described as follows:

$$p(x) = max\{0, z_x\}, z_x \le x, z_x \in \Phi_x^C(W_0), x < W_0, p(W_0) = W_0 \quad (\mathbf{10})$$

$$q(y) = max\{0, z_y\}, z_y \le y, z_y \in \Phi_y^C(H_0), y < H_0, q(H_0) = H_0 \quad (\mathbf{11})$$

$p(x)$ represents the cut position nearest to the sub-block width x, and correspondingly, $q(y)$ is the cut position nearest to the sub-block height y.



(a) Normalized cutting pattern (b) Non-normalized cutting pattern

Fig. 4. Normalized and Non-normalized cutting pattern (shaded parts are scrap)

Based on the above recursive function, we design the following Solver to solve the subproblem C-block.

Here is the description of the Solver：

Solver: Algorithm for solving 2D_UG_SLOPP

Input: Subproblem $R=(x, y)$, $\Phi_x^C(x)$, $\Phi_v^C(y)$
Output: Cutting Pattern recorded as $F(x, y)$

```
1  If (R=(x, y) has been solved)  Then
2   Return F(x, y);
3   Else If (x<w0 or y<h0)
4    Return 0;
5    Else
6     F*(x, y)=0;
7     For(zx∈Φx^C(x), 1≤zx≤x/2)
8      F*(x, y) ← max(Solver(R=(zx, y)) + Solver (R=(p(x-zx), y)), F*(x, y));
9      End for
```

```
10    F**(x, y)=0;
11   For(z_y∈Φ_y^C(y), 1≤z_y≤y/2)
12    F**(x, y) ← max(Solver(R=(x, z_y)) + Solver (R=(x, q(y-z_y))), F**(x, y));
13    End for
14   Return F(x, y)=max(g(x, y), F*(x, y), F**(x, y))
15  End if
16 End if
```

Obviously, because the original problem 2D_UG_SLOPP_D has defects, and the Solver is only called in the process of solving the original problem, which means that neither the x nor y in the subproblem $R=(x, y)$ is known in the original problem, the input here is not the initial value of the problem.

The equation (12) is used to determine the recursive function of the optimal cutting pattern with the D-block $R=(ox, oy, x, y)$:

$$F(ox,oy,x,y)=\begin{cases} F(x,y), \quad if\ R=(ox,oy,x,y)\ is\ C-block; \\ max\left\{\begin{array}{c} F(ox,oy,z_x,y)+F(ox+z_x,oy,x-z_x,y), \\ F(ox,oy,x-z_x,y)+F(ox+x-z_x,oy,z_x,y), \\ F\left(ox,oy,x,z_y\right)+F\left(ox,oy+z_y,x,y-z_y\right), \\ F\left(ox,oy,x,y-z_y\right)+F\left(ox,oy+y-z_y,x,z_y\right), \\ z_x\in\Phi_x^D(x), z_y\in\Phi_y^D(y), \\ w_0\le x\le W_0, h_0\le y\le H_0, \\ 0\le ox\le W_0, 0\le oy\le H_0, \end{array}\right\}, \\ if\ R=(ox,oy,x,y)\ is\ D-block. \end{cases} \tag{12}$$

Where, if $x<w_0\ or\ y<h_0$, then F(ox, oy, x, y).

The algorithm in this paper adopts denormalization strategy to solve the problem 2D_UG_SLOPP_D. In fact, the locations of the defects on D-block are uncertain, which means that this kind of normalization treatment may waste some plates, thus reducing the chance of obtaining the optimal solution. For example, if the defect is located in the lower left corner of a D-block, the more the upper right area of the block is used, the better the solution is possible. Based on all these favorable practices, we develop the advanced algorithm of predecessors [9, 10, 18] and get the improved heuristic dynamic programming algorithm IHDP.

Here is the description of the IHDP:

IHDP: Algorithm for solving 2D_UG_SLOPP_D

Input: Subpoblem $R=(ox, oy, x, y)$, $\Phi_x^D(x)$, $\Phi_y^D(y)$, Defects set $\{d_1, d_2, \ldots, d_n\}$

Output: Cutting pattern, Recorded as $F(ox, oy, x, y)$

```
1  If (R=(ox, oy, x, y) has been solved)  Then
2   Return F(ox, oy, x, y);
3   Else If (x<w_0 or y<h_0)
4    Return F(ox, oy, x, y)=0;
5    Else If (R=(ox, oy, x, y) is C-block)
6     Return F(x, y)=Solver(R=(x, y));
7     Else   // R=(ox, oy, x, y) is D-block
8      F*(ox, oy, x, y)=0;
9      For(z_x∈ Φ_x^D(x), 1≤z_x≤x-1)
10      F(1)(ox, oy, x, y)=IHDP(R=(ox, oy, z_x, y)+IHDP(R=(ox+z_x, oy, x-z_x, y);
11      F(2)(ox, oy, x, y)=IHDP(R=(ox, oy, x-z_x, y)+IHDP(R=(ox+x-z_x, oy, z_x, y);
12      F*(ox, oy, x, y)← max(F(1)(ox, oy, x, y), F(2)(ox, oy, x, y), F*(ox, oy, x, y));
```

```
13      End for
14      F**(ox, oy, x, y)=0;
15      For(z_y∈ Φ_y^D(y), 1≤z_y≤y-1)
16        F(3)(ox, oy, x, y)=IHDP(R=(ox, oy, x, z_y)+IHDP(R= (ox, oy+z_y, x, y-z_y);
17        F(4)(ox, oy, x, y)=IHDP(R=(ox, oy, x, y-z_y) +IHDP(R= (ox, oy+y-z_y, x, z_y);
18        F**(ox, oy, x, y) ← max(F(3)(ox, oy, x, y), F(4)(ox, oy, x, y), F**(ox, oy, x, y));
19       End for
20       Return F(ox, oy, x, y)= max(F*(ox, oy, x, y), F**(ox, oy, x, y));
21     End if
22   End if
23 End if
```

IHDP is implemented to solve the original problem 2D_UG_SLOPP_D and the subproblems with defects. Only when the subproblem is of type 2D_UG_ SLOPP, the Solver is called in line 6 to solve it. So, the initial values of the algorithm are R_0 = (0, 0, W_0, H_0), $\Phi_x^D(W_0)$, $\Phi_y^D(H_0)$ and all the defects d_1, d_2, …, d_n.

3.4. Algorithm complexity

In this section, we study the computational aspects of the algorithm. We analyze the time complexity in the worst case and get an estimation of pseudo polynomials.

Theorem 1 The time complexity in the worst case of the improved heuristic dynamic programming for solving the 2D_UG_SLOPP_D is:

$$O\left(|\Phi_x^D(W_0)| \cdot \left|\Phi_y^D(H_0)\right| \cdot \left(|\Phi_x^D(W_0)| + \left|\Phi_y^D(H_0)\right|\right)\right) \quad (\mathbf{13})$$

Proof For a given single large object (W_0, H_0), the recursive function requires $O(|\Phi_x^D(W_0)| \cdot \left|\Phi_y^D(H_0)\right|)$ operations for each iteration. Therefore, the calculation involves a total of time complexity as equation (13).

Theorem 2 Let $w_0 = \min\{w_i^s\}$ and $h_0 = \min\{h_i^s\}, i = 1,2,\dots,m$. And let $\rho_w = \lfloor W_0/w_0 \rfloor \le W_0/w_0, \rho_h = \lfloor H_0/h_0 \rfloor \le H_0/h_0$, then:

$$|\Phi_x^D(W_0)| = \sum_{t=1}^{\rho_w} C_{t+m-1}^t + 2n \quad (\mathbf{14})$$

$$\left|\Phi_y^D(H_0)\right| = \sum_{t=1}^{\rho_h} C_{t+m-1}^t + 2n \quad (\mathbf{15})$$

$$C_{t+m-1}^t = \frac{(t+m-1)!}{t!\,(m-1)!} \quad (\mathbf{16})$$

Proof By definition, each element in $\Phi_x^D(W_0)$ is a viable combination of the length of a small rectangular block $\sum_{t=1}^{\rho_w}(\sum_{i=1}^m w_i^s)^t \le W_0$ calculates the same structure of the number of terms in the polynomial. In order to obtain $\sum_{i=1}^m \alpha_i w_i^s$, in the above polynomial, t must take 1 toρ_w. That's because if $t > \rho_w$, then there is$\sum_{i=1}^m \alpha_i w_i^s > W_0$.

And $\sum_{t=1}^{\rho_w}(\sum_{i=1}^{m} w_i^s)^t = \sum_{t=1}^{\rho_w} C_{t+m-1}^t$. Add the above function (14) to the left and right edges of defects; similarly, the same is true for the function (15).

4. Calculation results and analysis

The algorithm used in this paper is implemented in the C/C++ programming language. The configuration of the computer used is: processor--Intel(R), Core (TM) i7cpu @360HZ, RAM 8GB, 64bit operator.

Three typical classes examples are included in this experiment. The first class is 14 instances in which 8 instances proposed by Carnieri et al. (1993) [8] include a single defect and 6 instances proposed by Vianna and Arenales (2006) [9] include multiple defects. The original object's width W_0 = 200, height H_0 = 100, and 5 types of small rectangular blocks. The other two classes are generated by Neidlein's [18] instance generator. One class of them has already been generated and adopted both by both Afsharian et al. [11] and by Martin et al. [12]. Another class is generated by ourselves, and the seed values of random numbers are 3, 7 and 11, respectively.

In this paper, five other typical algorithms are selected to compare their effectiveness and efficiency with IHDP. The **o**bjective **f**unction **v**alues (OFV) obtained by all these algorithms is the important index. Furthermore, according to the literature [18], the algorithm DPC (dynamic programming with complete discretization set) is implemented. Another index, GAP = $(OFV_{DPC}\text{-}OFV_{IHDP})/ (OFV_{DPC} + 10^{-10})*100$, is used to present these algorithms' performance.

4.1. International Samples

In Table 2, The algorithm is compared with other three algorithms. Neidlein et al. (2008) [10] only computes the instances with a single defect and obtains the optimal solutions of 5 out of 8 instances. Vianna and Arenales [9] gets an optimal solution of the instance with multiple defects, however, it does not public their computation time. Both Afsharian et al. [11] and IHDP obtain the 14 optimal solutions in a short time.

Table 2. Operation results and comparison of four algorithms

Ins.	IHDP		Afsharian etc.(2014)		Vianna etc.(2006)		Neidlein etc.(2008)	
	OFV	Comp. Time(s)	OFV	Comp. Time(s)	OFV	Comp. Time(s)	OFV	Comp. Time(s)
A1	**166**	0.86	**166**	18.86	**166**	4.61	**166**	0.52
A2	**166**	0.67	**166**	16.43	160	3.57	160	0.77
A3	**166**	0.74	**166**	16.47	162	4.40	162	1.77
A4	**164**	0.25	**164**	18.25	160	3.15	160	0.27
A5	**164**	0.31	**164**	76.96	**164**	13.51	**164**	4.11
A6	**164**	0.47	**164**	0.90	**164**	1.32	**164**	1.44
A7	**158**	0.21	**158**	0.81	**158**	12.47	**158**	1.07
A8	**154**	0.13	**154**	1.21	**154**	8.07	**154**	0.50
A9	**160**	0.87	**160**	14.32	153	-	-	-
A10	**158**	0.49	**158**	2.22	148	-	-	-
A11	**151**	6.29	**151**	26.78	143	-	-	-

A12	**156**	119.30	**156**	1126.44	150	-	-	-
A13	**150**	2.19	**150**	9.06	142	-	-	-
A14	**160**	0.09	**160**	1.00	**160**	-	-	-

Note: Bold type in the table represents the optimal solution

In addition, the optimal layout the two instances A11 and A12 which have 4 and 5 defects respectively are given below (see Fig. 5-6).

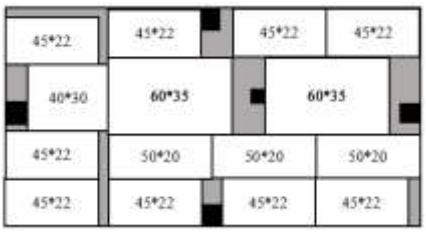


Fig. 5. A11 cutting pattern result

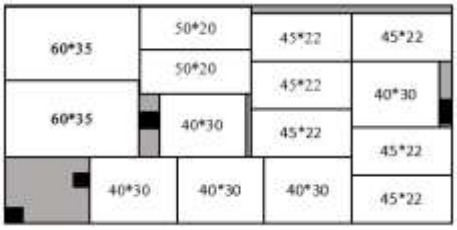


Fig. 6. A12 cutting pattern result

4.2. Randomly generated instances

These examples are set by the instance generator procedure of Neidlein and Wäscher (2016) [18]. With respect to the size of the large plate, it is distinguished between two categories of problem instances, i.e., quadratic and non-quadratic. In small category, the size of the large object is fixed to 5,625 square units, and the size of the large object is fixed to 22,500 square units in medium category. Furthermore, the two different shapes of the large plate considered for each category are quadratic (75, 75) and non-quadratic (112, 50) for the small category and quadratic (150, 150) and non-quadratic (225, 100) for medium category, respectively. The number of types of small rectangles is set to 5. 10, 15, 20 and 25. The type width and height of the small rectangular block are uniformly obtained from $[W_0/\varpi, 3W_0/4]$ and $[H_0/\varpi, 3H_0/4]$, respectively, where ϖ in all categories is 6, 8, and 10. The defect is set to 1~4, and 15 examples of each defect are averaged for a group. The width and height of the defect are uniformly obtained from the ranges $[W_0/10, W_0/6]$ and $[H_0/10, H_0/6]$, respectively. The position of each defect is represented by the position of the defect in the lower left corner of the large rectangular block, generated using a uniform distribution in the range of $[0, W_0 - w_0]$ and $[0, H_0 - h_0]$, and then these values are rounded. So, the instances number of each size category are 2×5×3×15×4=1800.

Instances in the literatures. The instances in Table 3 and Table 4 are generated by Afs -harian et al. (2014) [11] without providing the value of their random seed, however, the data can be obtained from the following website: www. dep.ufscar.br/docentes/munari/ cuttingpacking/.

In Table 3, the other three algorithms are the best ones we have got. As a complete DP (Dynamic programming takes every integer both in [1, W_0] and in [1, H_0] as cutting point) [18], DPC does obtain the optimal solutions of all instances in guillotine manner. IHDP obtains the optimal solution of all cases, however, DPD and B&BC get the optimal solutions of one group and ten groups of instances, respectively. However, The

Gap of B&BC on the average of 30 groups is more than twice that of DPD, so the stability of the latter is better than the former. B&BC takes far more time than the other three algorithms.

In Table 4, Martin et al. [12] doesn't show the calculation details of the 30 group instances, so the computation results of DPC, DPD and IHDP are compared. None of the optimal solutions of these instances are obtained by DPD. However, IHDP gets 26 out of 30 groups optimal solutions of the instances, and the four GAPs are very close to 0.00. Here, although the computation time difference among them is not very big, IHDP takes more time than DPD, but less than DPC.

New randomly generated instances. In order to further test the performance of IHDP, t he generator of Neidlein et al. (2016) [18] is used to generate new data. Table 5 and Tab le 6 are instances of small and medium scale, and the value of their random number seed =3. Table 7 and Table 8 are small scale instances, and the value of their random number seed=7, 11, respectively (randomly selected from the uniform distribution of [1, 12]). B ecause these are new data, only DPC and IHDP are used for comparison.

Table 5 shows the computation results of small-scale instances. Both IHDP and DPC get the optimal solution of all cases, but IHDP takes almost a tenth of DPC. Table 6 shows the computation results of the medium scale case. IHDP does not get the optimal solution on five groups of instances. However, the result of IHDP is very close to the optimal solution, and its final average Gap is still 0.000. Moreover, it takes less than half the computing time of DPC.

Both in Table 7 and Table 8, 1800 small scale instances are generated by seed values of 7 and 11, respectively. In Table 7, IHDP obtains the optimal solutions of 28 out of 30 groups of the instances, but the GAP is only 0.2%. In Table 8, IHDP obtains all the optimal solution of the instances. In both cases, IHDP takes about half as long as DPC.

According to the results shown in Tables 3 to 8 above, IHDP performs stablely on the instances generated by both ourselves and Afsharian et al. [11], and it is an effective and efficient algorithm to solve the original problem 2D_UG_SLOPP_D in guillotine mode.

5. Conclusion

In this paper, the smaller discretization sets are constructed to solve the two-dimensional cutting problem with defects in guillotine manner. Especially, an improved heuristic-dynamic programming algorithm is proposed, which adapts two different methods to for the subproblems 2D_UG_SLOPP_D and 2D_UG_SLOPP, respectively. Almost all the optimal solutions of over ten thousand typical instances are obtained. An important theorem on its complexity of the algorithm is proved. Future research could focus on solving the larger scale instance or on modifying the discretization set definition or on solving different type of cutting problem such as involving the constraints of the largest number of each type of the small rectangular block.

Table 3. Four algorithms comparison on the small size 1800 instances of Afsharian et al.[11]

instances	m	ϖ	DPC		DPD			B&BC			IHDP		
			OFV	time(s)	OFV	time(s)	GAP(%)	OFV	time(s)	GAP(%)	OFV	time(s)	GAP(%)
1	5	6	**3694.31**	11.45	**3694.31**	3.52	0.00	**3694.31**	0.05	0.00	**3694.31**	0.41	0.00
2	10	6	**4256.01**	12.12	4229.85	4.52	0.61	**4256.01**	15.20	0.00	**4256.01**	2.50	0.00
3	15	6	**4566.10**	13.33	4540.38	5.36	0.56	**4566.10**	1.57	0.00	**4566.10**	3.92	0.00
4	20	6	**4615.43**	13.09	4593.41	5.76	0.48	4605.70	42.80	0.21	**4615.43**	5.32	0.00
5	25	6	**4694.01**	13.89	4674.66	6.19	0.41	4691.43	56.70	0.05	**4694.01**	6.31	0.00
6	5	8	**3683.23**	11.14	3638.03	2.86	1.23	**3683.23**	0.87	0.00	**3683.23**	0.60	0.00
7	10	8	**4386.16**	12.94	4375.50	4.74	0.24	4372.03	62.00	0.32	**4386.16**	3.20	0.00
8	15	8	**4613.23**	14.07	4585.26	6.09	0.61	4604.15	78.03	0.19	**4613.23**	5.26	0.00
9	20	8	**4883.78**	14.69	4876.38	6.27	0.15	4850.66	150.20	0.67	**4883.78**	6.83	0.00
10	25	8	**4835.60**	14.48	4826.93	7.76	0.18	4792.66	91.92	0.88	**4835.60**	7.49	0.00
11	5	10	**4083.56**	12.71	4055.10	3.18	0.70	4001.58	16.56	2.00	**4083.56**	1.11	0.00
12	10	10	**4710.91**	14.60	4686.56	5.69	0.52	4637.40	107.65	1.56	**4710.91**	5.39	0.00
13	15	10	**4845.16**	14.78	4826.68	6.74	0.38	4678.93	122.56	3.43	**4845.16**	6.25	0.00
14	20	10	**4928.65**	15.25	4920.95	9.35	0.16	4629.06	155.83	6.07	**4928.65**	8.20	0.00
15	25	10	**4968.50**	16.13	4960.23	10.71	0.17	4783.45	217.93	3.72	**4968.50**	9.44	0.00
16	5	6	**3530.41**	10.74	3480.63	2.61	1.41	**3530.41**	0.03	0.00	**3530.41**	0.26	0.00
17	10	6	**4339.18**	11.53	4305.43	4.31	0.78	**4339.18**	20.38	0.00	**4339.18**	2.45	0.00
18	15	6	**4495.20**	14.54	4470.76	4.31	0.54	**4495.20**	13.40	0.00	**4495.20**	4.05	0.00
19	20	6	**4618.16**	14.66	4607.75	5.80	0.23	4611.68	59.53	0.14	**4618.16**	5.77	0.00
20	25	6	**4686.21**	15.42	4679.73	6.78	0.14	4671.68	82.28	0.30	**4686.21**	6.75	0.00
21	5	8	**3829.66**	13.69	3752.31	2.03	2.02	**3829.66**	7.83	0.00	**3829.66**	0.68	0.00
22	10	8	**4536.90**	15.30	4523.00	4.56	0.31	**4536.90**	26.33	0.00	**4536.90**	4.02	0.00
23	15	8	**4792.70**	14.54	4766.90	5.65	0.54	4770.53	130.25	0.46	**4792.70**	6.10	0.00
24	20	8	**4706.16**	15.16	4685.91	5.92	0.43	4681.53	74.36	0.52	**4706.16**	6.78	0.00
25	25	8	**4791.00**	15.66	4772.01	8.09	0.40	4715.31	121.96	1.57	**4791.00**	7.92	0.00
26	5	10	**4138.25**	13.43	4076.48	3.63	1.49	**4138.25**	11.50	0.00	**4138.25**	0.99	0.00
27	10	10	**4396.10**	15.29	4357.55	4.53	0.88	4374.31	70.63	0.49	**4396.10**	3.94	0.00
28	15	10	**4688.12**	15.95	4661.92	5.70	0.56	4590.73	149.97	2.07	**4688.12**	6.31	0.00
29	20	10	**4895.85**	16.14	4891.43	8.71	0.09	4705.36	154.61	3.89	**4895.85**	8.40	0.00
30	25	10	**4988.35**	16.20	4975.15	9.31	0.26	4856.01	206.26	2.65	**4988.35**	8.92	0.00
Average			4506.56	14.10	4483.04	5.69	0.52	4456.45	74.97	1.11	4506.56	4.85	0.00

Note: Bold type in the table represents the optimal solution

Table 4. Comparewith DPC on the medium size 1800 instances of Afsharian et al.[11]

instances	m	ϖ	DPC		DPD			IHDP		
			OFV	time(s)	OFV	time(s)	GAP(%)	OFV	time(s)	GAP(%)
1	5	6	14794.82	209.58	**14546.66**	24.723	1.677	14794.82	7.89	0.000
2	10	6	15772.18	202.97	**15544.76**	35.661	1.442	15772.18	42.14	0.000
3	15	6	17356.57	146.04	**17263.78**	35.827	0.535	17356.57	77.21	0.000
4	20	6	18256.58	160.21	**18128.60**	58.636	0.701	18256.58	138.48	0.000
5	25	6	18824.05	172.32	**18716.65**	69.199	0.571	**18701.42**	163.10	**0.007**
6	5	8	14384.88	183.29	**14276.08**	27.278	0.756	14384.88	4.94	0.000
7	10	8	17780.28	280.81	**17632.83**	40.890	0.829	17780.28	74.94	0.000
8	15	8	18747.35	182.60	**18615.33**	72.122	0.704	18747.35	136.41	0.000
9	20	8	19315.80	193.07	**19193.25**	73.607	0.634	19315.80	185.78	0.000
10	25	8	19530.53	189.00	**19435.88**	131.198	0.485	19530.53	184.81	0.000
11	5	10	15174.78	237.29	**15067.86**	24.038	0.705	15174.78	9.19	0.000
12	10	10	18839.87	184.59	**18775.98**	63.492	0.339	18839.87	116.96	0.000
13	15	10	18747.58	192.31	**18685.03**	71.939	0.334	**18611.28**	153.12	**0.007**
14	20	10	19438.30	193.51	**19372.35**	99.747	0.339	19438.30	187.32	0.000
15	25	10	19822.68	217.56	**19724.56**	137.128	0.495	19822.68	216.96	0.000
16	5	6	13611.92	100.51	**13420.90**	21.29	1.403	13611.92	6.08	0.000
17	10	6	16938.10	152.79	**16709.06**	43.03	1.352	16938.10	84.64	0.000
18	15	6	17203.60	151.04	**17058.16**	45.42	0.845	17203.60	96.32	0.000
19	20	6	18606.00	181.05	**18475.80**	62.96	0.700	18606.00	141.73	0.000
20	25	6	18708.25	194.25	**18591.30**	74.73	0.625	18708.25	190.35	0.000
21	5	8	15217.80	128.29	**15024.26**	23.33	1.272	15217.80	15.61	0.000
22	10	8	17720.82	177.21	**17541.91**	53.70	1.010	17720.82	163.48	0.000
23	15	8	18620.30	196.23	**18491.06**	63.04	0.694	**18518.05**	143.48	**0.005**
24	20	8	18999.55	212.83	**18814.46**	84.30	0.974	18999.55	210.44	0.000
25	25	8	19382.40	225.39	**19215.23**	109.88	0.862	19382.40	217.22	0.000
26	5	10	15542.52	138.11	**15248.20**	35.21	1.894	15542.52	31.84	0.000
27	10	10	17366.72	179.47	**17191.66**	47.83	1.008	17366.72	106.46	0.000
28	15	10	18913.72	203.36	**18733.61**	92.64	0.952	18913.72	192.05	0.000
29	20	10	19632.15	220.12	**19537.85**	131.77	0.480	**19513.54**	220.01	**0.006**
30	25	10	19707.63	233.18	**19561.21**	122.08	0.743	19707.63	230.33	0.000
Average			17765.26	187.97	17619.81	65.89	0.845	17749.26	124.98	0.001

Note: Bold type in the table represents the non-optimal solution

Table 5. Compare with DPC on the small size 1800 new instances (seed=3)

instances	m	ϖ	DPC		IHDP		
			OFV	time(s)	OFV	time(s)	GAP(%)
1	5	6	3825.80	9.82	3825.80	0.15	0.00
2	10	6	4381.25	12.16	4381.25	0.62	0.00
3	15	6	4567.50	12.53	4567.50	0.78	0.00
4	20	6	4726.13	12.66	4726.13	1.47	0.00
5	25	6	4842.80	12.19	4842.80	1.55	0.00
6	5	8	3997.50	10.76	3997.50	0.19	0.00
7	10	8	4630.67	12.94	4630.67	1.03	0.00
8	15	8	4793.17	13.18	4793.17	1.30	0.00
9	20	8	4863.02	14.21	4863.02	1.71	0.00
10	25	8	5047.80	14.21	5047.80	2.05	0.00
11	5	10	4166.97	11.44	4166.97	0.24	0.00
12	10	10	4707.54	13.31	4707.54	1.17	0.00
13	15	10	4864.12	13.45	4864.12	1.45	0.00
14	20	10	4981.47	14.16	4981.47	2.08	0.00
15	25	10	5099.68	14.56	5099.68	2.61	0.00
16	5	6	3700.47	11.51	3700.47	0.13	0.00
17	10	6	4366.59	13.13	4366.59	0.57	0.00
18	15	6	4504.84	12.93	4504.84	0.88	0.00
19	20	6	4600.78	13.74	4600.78	0.89	0.00
20	25	6	4697.32	13.88	4697.32	1.43	0.00
21	5	8	3838.69	10.56	3838.69	0.15	0.00
22	10	8	4560.90	13.47	4560.90	0.75	0.00
23	15	8	4769.70	13.89	4769.70	1.21	0.00
24	20	8	4826.15	14.88	4826.15	1.67	0.00
25	25	8	4944.00	14.69	4944.00	2.05	0.00
26	5	10	4033.33	11.52	4033.33	0.15	0.00
27	10	10	4562.67	14.69	4562.67	0.89	0.00
28	15	10	4781.54	15.20	4781.54	1.60	0.00
29	20	10	4959.79	14.43	4959.79	2..06	0.00
30	25	10	5014.05	15.24	5014.05	2.31	0.00
Average			4588.54	13.18	4588.54	1.10	0.00

Table 6. Compare with DPC on the medium size 1800 new instances (seed=3)

instances	m	ϖ	DPC		IHDP		
			OFV	time(s)	OFV	time(s)	GAP(%)
1	5	6	14974.25	277.85	**14972.15**	4.51	**0.000**
2	10	6	17113.60	290.08	17113.60	56.95	0.000
3	15	6	18259.88	278.21	18259.88	112.63	0.000
4	20	6	18250.43	310.59	18250.43	138.08	0.000
5	25	6	18769.08	282.40	18769.08	155.32	0.000
6	5	8	15993.28	293.52	15993.28	8.45	0.000
7	10	8	18022.70	283.77	18022.70	87.99	0.000
8	15	8	18772.30	274.74	18772.30	143.05	0.000
9	20	8	19341.68	281.89	19341.68	192.06	0.000
10	25	8	19653.10	287.14	19653.10	227.91	0.000
11	5	10	15902.78	288.95	15902.78	7.68	0.000
12	10	10	18600.38	275.67	18600.38	99.46	0.000
13	15	10	19213.53	275.63	**19213.10**	167.44	**0.000**
14	20	10	19730.58	272.70	19730.58	207.27	0.000
15	25	10	20012.65	282.42	20012.65	260.77	0.000
16	5	6	14730.78	324.04	**14623.88**	5.34	**0.007**
17	10	6	17040.13	301.38	17040.13	47.19	0.000
18	15	6	18025.05	305.29	18025.05	103.17	0.000
19	20	6	18465.53	316.33	18465.53	153.18	0.000

20	25	6	18842.85	313.16	18842.85	189.40	0.000
21	5	8	15830.78	304.80	**15827.15**	6.76	0.000
22	10	8	17801.28	317.94	17801.28	81.35	0.000
23	15	8	18732.18	299.69	18732.18	147.56	0.000
24	20	8	19156.50	313.54	19156.50	186.86	0.000
25	25	8	19384.78	306.00	19384.78	216.96	0.000
26	5	10	15078.53	297.44	15078.53	6.96	0.000
27	10	10	17974.48	313.68	**17958.15**	90.66	**0.001**
28	15	10	18850.90	310.60	18850.90	166.48	0.000
29	20	10	19401.13	311.13	19401.13	232.41	0.000
30	25	10	20059.15	308.62	20059.15	274.27	0.000
Average			18066.14	296.64	18061.83	125.94	0.000

Note: Bold type in the table represents the non-optimal solution

Table 7. Compare with DPC on the small size 1800 new instances (seed=7)

Ins.	m	ϖ	DPC		IHDP		
			OFV	time(s)	OFV	time(s)	GAP(%)
1	5	6	3947.23	2.17	3947.23	0.12	0.00
2	10	6	4388.65	3.11	4388.65	0.85	0.00
3	15	6	4526.98	3.57	4526.98	1.75	0.00
4	20	6	4654.13	3.89	4654.13	2.23	0.00
5	25	6	4758.02	4.21	4758.02	2.31	0.00
6	5	8	4020.60	2.42	4020.60	0.24	0.00
7	10	8	4507.75	3.99	4507.75	1.44	0.00
8	15	8	4732.47	4.66	4732.47	3.27	0.00
9	20	8	4854.50	5.15	4854.50	3.62	0.00
10	25	8	4924.37	5.41	4924.37	4.56	0.00
11	5	10	4105.08	3.45	4105.08	0.93	0.00
12	10	10	4732.95	5.45	4732.95	3.31	0.00
13	15	10	4899.15	6.25	4899.15	5.70	0.00
14	20	10	5006.60	6.72	5006.60	5.69	0.00
15	25	10	5063.02	6.83	5063.02	7.38	0.00
16	5	6	3810.30	4.33	**3808.07**	0.20	0.06
17	10	6	4303.37	5.14	4303.37	0.63	0.00
18	15	6	4529.53	5.56	4529.53	0.97	0.00
19	20	6	4598.10	5.92	4598.10	1.42	0.00
20	25	6	4702.70	6.22	4702.70	2.02	0.00
21	5	8	4041.78	4.88	4041.78	0.74	0.00
22	10	8	4501.80	5.97	4501.80	2.29	0.00
23	15	8	4732.78	6.74	4732.78	2.96	0.00
24	20	8	4803.43	6.82	4803.43	3.55	0.00
25	25	8	4851.45	7.02	4851.45	3.88	0.00
26	5	10	4055.08	5.12	4055.08	0.73	0.00
27	10	10	4730.05	6.77	**4729.92**	2.95	0.003
28	15	10	4820.05	7.01	4820.05	3.91	0.00
29	20	10	4933.32	7.40	4933.32	5.06	0.00
30	25	10	4992.20	8.48	4992.20	6.52	0.00
Average			4584.25	5.36	4584.17	2.71	0.002

Note: Bold type in the table represents the non-optimal solution

Table 8. Compare with DPC on the small size 1800 new instances (seed=11)

Ins.	*m*	ϖ	DPC		IHDP		
			OFV	time(s)	OFV	time(s)	GAP(%)
1	5	6	3854.90	2.34	3854.90	0.18	0.00
2	10	6	4352.48	3.46	4352.48	0.90	0.00
3	15	6	4548.65	4.18	4555.65	1.36	0.00
4	20	6	4658.13	4.49	4658.13	1.87	0.00
5	25	6	4761.37	4.77	4761.37	2.43	0.00
6	5	8	3918.62	2.87	3918.62	0.34	0.00
7	10	8	4538.45	4.77	4538.45	1.53	0.00
8	15	8	4758.63	5.67	4765.73	2.55	0.00
9	20	8	4798.27	5.45	4798.27	3.02	0.00
10	25	8	4926.80	6.39	4926.80	3.99	0.00
11	5	10	3963.83	3.34	3963.83	0.69	0.00
12	10	10	4635.02	5.56	4635.02	2.33	0.00
13	15	10	4851.70	6.54	4851.70	4.07	0.00
14	20	10	4926.92	6.91	4926.92	4.80	0.00
15	25	10	5016.78	7.57	5016.78	6.01	0.00
16	5	6	3818.85	2.97	3818.85	0.15	0.00
17	10	6	4257.70	3.89	4257.70	0.61	0.00
18	15	6	4586.68	5.02	4586.68	1.46	0.00
19	20	6	4649.18	5.39	4649.18	2.09	0.00
20	25	6	4776.97	5.86	4776.97	2.43	0.00
21	5	8	3955.52	3.69	3955.52	0.33	0.00
22	10	8	4524.15	5.00	4524.15	1.44	0.00
23	15	8	4716.22	6.12	4716.22	2.62	0.00
24	20	8	4795.03	6.08	4795.03	3.41	0.00
25	25	8	4884.32	6.81	4884.32	4.28	0.00
26	5	10	4100.58	4.31	4100.58	0.58	0.00
27	10	10	4610.17	5.85	4610.17	2.21	0.00
28	15	10	4806.98	6.36	4806.98	3.37	0.00
29	20	10	4908.53	7.38	4908.53	4.58	0.00
30	25	10	4982.80	8.07	4982.80	5.12	0.00
Average			4562.81	5.24	4562.81	2.36	0.00

Acknowledgements. This work was supported by the National Natural Science Foundation of China (Grant Nos.61862027, 61702238 and 61866014), the Natural Science Foundation Project of Jiangxi (Grant No.20192BAB207008), the Science Foundation of Educational Commission of Jiangxi Province (Grant Nos.Gjj170316 and Gjj180264).

References

1. Wäscher G, Haußner H, Schumann H. An improved typology of cutting and packing problems [J]. European Journal of Operational Research, 2007, 183(3): 1109-1130.
2. Wang L, Liu Q, Chen X. Heuristic search algorithm for the rectangular fixed-size guillotine bin packing problem [J]. Journal of Software, 2017, 28: 1640-1654.

3. Song X, Chu C B, Lewis R, et al. A worest case analysis of a dynamic programming-based heuristic algorithm for 2D unconstrained guillotine cutting [J]. European Journal of Operational Research, 2010, 202(2): 368-378.
4. Wuttke D A, Heese H S. Two-dimensional cutting stock problem with sequence dependent setup times[J]. European Journal of Operational Research, 2017, 265: 303-315.
5. Yoon K, Ahn S, Kang M. An improved best-first branch-and-bound algorithm for constrained two-dimensional guillotine cutting problems [J]. International Journal of Production Research, 2013, 51(6): 1680-1693.
6. Herz, J. C. Recursive computational procedure for two-dimensional stock cutting [J]. IBM Journal of Research and Development, 1972, 16(5): 462-469.
7. Beasley, J. E. Algorithms for unconstrained two-dimensional guillotine cutting [J]. Journal of the Operational Research Society, 1985(a), 36(4): 297-306.
8. Carnieri C, Mendoza G A, Luppold W G. Optimal cutting of dimension parts from lumber with a defect: A heuristic solution procedure [J]. Forest Products Journal, 1993, 43: 66-72.
9. Vianna ACG, Arenales MN. Problema de corte de placas defeituosas. Pesqui Operacional, 2006, 26: 185-202.
10. Neidlein V, Vianna, A C G, Arenales, M.N, Wäscher, G. The two-dimensional guillotine-layout cutting problem with a single defect - An and/or-graph approach [J]. Operations Research Proceedings 2008 (Eds.: Fleischmann, B. et al.). Berlin, Heidelberg, Springer-Verlag, 85-90.
11. Afsharian M, Niknejad A, Wäscher G. A heuristic, dynamic programming-based approach for a two-dimensional cutting problem with defects [J]. OR Spectrum, 2014, 36(4):971-999.
12. Martin M, Hokama Pedro H D B, Morabito R, Munari, P. The constrained two-dimensional guillotine cutting problem with defects: an ILP formulation, a Benders decomposition and a CP-based algorithm[J]. International Journal of Production Research, 2019, 1-18.
13. Gonçalves J F, Wäscher G. A MIP model and a biased random-key genetic algorithm based approach for a two-dimensional cutting problem with defects. European Journal of Operational Research, 2020, preprint. doi: https://doi.org/10.1016/j.ejor.2020.04.028
14. Birgin E G, Romão, O. C, Ronconi D P. The multiperiod two-dimensional non-guillotine cutting stock problem with usable leftovers[J]. International Transactions in Operational Research, 2019, 1392-1418.
15. Velasco A S, Eduardo U. Improved state space relaxation for constrained two-dimensional guillotine cutting problems[J]. European Journal of Operational Research, 2019, 272: 106-120.
16. Russo M, Boccia M, Sforza A, Sterle C. Constrained two-dimensional guillotine cutting problem: upper-bound review and categorization. International Transactions in Operation Research, 2020, 27: 794-834.
17. Wu K, Min X, Zhang D. Research on two-dimensional cutting problem with defects. 2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS), Beijing, China: 506-511.
18. Neidlein V, Wäscher G. SLOPPGEN: a problem generator for the two-dimensional rectangular single large object placement problem. International Transactions in Operational Research, 2016, 23:173-186.

Aihua YIN is a Senior Researcher at Jiangxi University of Finance and Economics. He received the Ph.D. degree in Computer Science from the Huazhong University of Science and Technology, Wuhan, China. His research interests include job shop scheduling problem, cutting and packing problem.

Chong CHEN is a Master degree candidate at Jiangxi University of Finance and Economics. He received the Bachelor degree in Computer Science and Technology from the Wuhan University of Science and Technology, China. His research interests include cutting problem.

Dongping HU is an associate professor at Jiangxi University of Finance and Economics. She received the Ph.D. degree in Information Security from the Huazhong University of Science and Technology, Wuhan, China. Her research focuses on the searchable encryption, the application intelligent algorithm in information security.

Jianghai HUANG is a Master degree candidate at Jiangxi University of Finance and Economics. He received the Bachelor degree in Software Engineering from the Tianjin University Renai College, Tianjin, China. His research interests include cutting problem.

Fan Yang is a Master degree candidate at Jiangxi University of Finance and Economics. She received the Bachelor degree in Electronic Commence from the Wuhan University of Science and Technology, Wuhan, China. Her research interests include Text Big Data Analysis.

Received: January 25, 2020; Accepted: July 02, 2020