Column Generation Algorithms for Two-dimensional Cutting Problem with Surface Defects

Qi Zhang, Yang Xing, Changtian Zhang, Xiaoxu Sun, Bin Hu, and Arup Das

***Abstract*—This work investigates a two-dimensional cutting problem arising from the cutting process of plate products, in which order plates of given sizes are to be cut from a finite number of mother plates that contain several surface defects. Placement of order plates on a mother plate is restricted by defects of varying severity located on the mother plate as well as the quality grades of order plates to be cut. We adopt a two-staged guillotine cutting mode, where the first-staged cutting position is determined by a width permutation scheme. Considering the allocation intervals of order plates, we first formulate the problem as a mixed-integer programming model that aims at maximizing the total revenue of a steel plant. Then, we present a column generation-based (CG-based) algorithm to solve it. To further improve its performance, we devise an accelerated CG (ACG) algorithm that embeds three heuristic accelerating strategies in the pricing process. At last, we test the proposed algorithms by a series of randomly generated instances, which are constructed according to actual production rules. The experimental results show that ACG algorithm can effectively solve large-sized instances.**

***Key Words*—Two-dimensional cutting, Surface defects, Quality grades, Two-staged guillotine cutting, Column generation.**

I. Introduction

IN modern steel manufacturing, the continuous casting–hot rolling (CC–HR) process is widely used for large-scale steel plate production. However, fluctuations in equipment status and variations in process parameters often result in surface defects on steel plates, including scratches, inclusions, warping, cracks, and embedded scale (see Fig. 1). According to the World Steel Association in 2023, approximately 8%–12% of steel plates exhibit surface defects, causing annual revenue losses of up to USD 180 million–250 million per medium-sized steel plant. Although previous studies have shown that about 60% of customer orders can tolerate a certain degree of defects, most steel mills currently treat defective mother plates as scrap, leading to significant resource waste and economic losses.

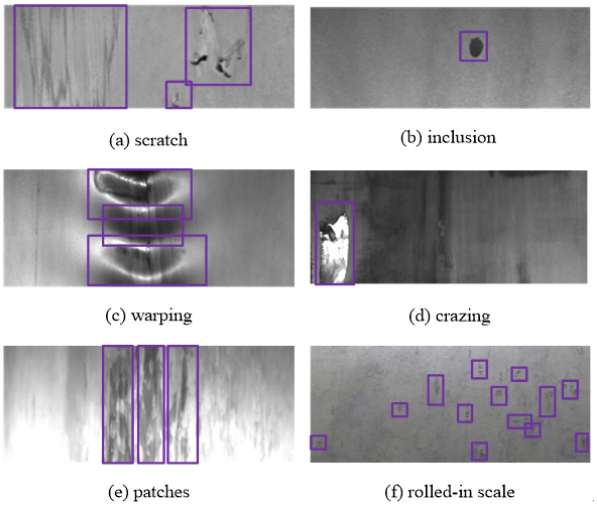


Fig. 1. Surface defects of varying severity.

To improve profitability, steel mills urgently need to optimize the utilization strategy of defective mother plates. In current industrial practice, a two-stage orthogonal guillotine cutting process is commonly used: first, double-side trimmers horizontally cut the mother plate into several ”panels”, and then a slitter performs vertical cuts on these panels to produce steel plates that meet customer specifications. This process must simultaneously address complex constraints, including overlapping rectangular defects and their severity levels, customer order tolerance for defect levels, geometric feasibility (plates must not overlap and must be aligned parallel to the edges of the mother plate), and heterogeneous sizes of multiple mother plates. This work formally defines the problem as the Two-Dimensional Guillotine Cutting + Multiple Mother Plates + Defect and Quality Constraints Layout Optimization Problem (2D_G_MHLOPP_DQ). This formulation extends the NP-hard 2D_SLOPP problem [1], [2], resulting in substantially increased computational complexity that traditional algorithms struggle to address effectively [3]–[5].

While Cutting and Packing (C&P) problems have been extensively studied across various domains, such as apparel, energy, and warehousing, existing approaches fall short when applied to the specific needs of the steel industry. These needs

Manuscript received June 27, 2025; revised July 14 and July 17, 2025; accepted July 23, 2025. This article was recommended for publication by Editor Shujin Qin upon evaluation of the reviewers' comments. This work was supported by Liaoning Province Education Department Scientific Research Foundation of China under Grant JYTQN2023366.

Q. Zhang is with the School of Information Engineering, Shenyang University of Chemical Technology, Shenyang 110142, China. (e-mail: qizhang@syuct.edu.cn).

Y. Xing, C. Zhang, and X. Sun are with the School of Information Engineering, Shenyang University of Chemical Technology, Shenyang 110142, China. (e-mail: xingyang0315@163.com, longsky2001@163.com, sunxiaoxu0829@163.com).

B. Hu is with the Department of Computer Science and Technology, Kean University, Union, NJ 07083, USA (e-mail: binhu.philip@gmail.com).

A. Das is with Jackson Lewis, USA (e-mail: aruprataneas2008@gmail.com).

Corresponding Author: Qi Zhang

TABLE I Comparison of representative studies and identified research gaps.

Study	**QG**	**Cutting Stage**	**MMP**	**Method Type**	**Research Gap / Limitation**
Parreño et al [10].	×	Three-stage	×	Beam Search	No quality grading support
Martin et al [11].	×	Unlimited stages	✓	Benders decomposition	No QG; unlimited stages increase complexity
Durak & Aksu [12]	✓	Unlimited stages	×	Online dynamic programming	Point defects only; no MMP; unlimited stages
Chen et al. [13]	✓	Single/two-stage	×	Deep Reinforcement Learning	High computational cost; no overlapping defects; no MMP
Wang & Zhang [14]	×	Multi-stage	✓	Improved heuristic algorithm	No QG; simple defect modeling; not two-stage
Our work	✓	Two-stage	✓	Accelerated Column Generation (ACG)	Fills gap: QG + 2S + MMP + overlapping defects

include handling defective plates, incorporating quality level constraints, and managing a two-stage cutting process. For instance, early work by Gilmore and Gomory [6] employed Dynamic Programming (DP) to address defect-aware cutting, but such methods are difficult to scale to industrial problem sizes. Later studies proposed various cutting strategies and modeling techniques; however, they often suffer from critical limitations: inability to accommodate overlapping defects, lack of support for quality grading, incompatibility with the industrial two-stage process, or failure to model multiple heterogeneous mother plates [7]–[9]. These limitations are summarized in Table I, which highlights the key features and research gaps of representative studies. As illustrated, no existing method simultaneously satisfies the full set of industrial requirements: two-stage cutting, overlapping defect handling, quality level constraints, and heterogeneous mother plates. This capability gap represents a critical technological bottleneck for steel manufacturers aiming to improve resource efficiency and economic performance.

While recent advances in column generation (CG) have improved defect handling in cutting problems, critical gaps persist for industrially relevant constraints. Silva et al. [15] proposed a CG approach for irregular defects using dynamic safe zones but excluded overlapping defects, which induce non-convex feasible regions. Similarly, Moreira et al. [16] developed a robust CG framework for quality-graded materials yet required defect-free subregions, limiting applicability to lumber or leather with pervasive flaws. For overlapping defect constraints, Caro et al. [17] noted that standard CG pricing models become NP-hard under intersecting exclusion zones—a challenge unresolved in contemporary literature. Recent surveys (Furini et al., [18]) highlight that no CG method simultaneously addresses: (i) overlapping defect aggregation, (ii) quality-driven cutting hierarchies, and (iii) real-time responsiveness for industrial-scale instances.

To address these challenges, this work adopts the Column Generation (CG) method as the core solution framework. CG is particularly well-suited for large-scale combinatorial optimization problems because it generates valuable columns (cutting patterns) dynamically, without requiring exhaustive enumeration of the pattern space. This makes it highly effective for managing complex cutting combinations while supporting intricate constraints such as multi-mother plate (MMP) handling and multi-quality grading (QG), thereby enabling global optimization and strong industrial applicability. However, standard CG approaches tend to suffer from slow convergence, especially in scenarios characterized by high combinatorial complexity, dense overlapping defects, and stringent quality requirements. To overcome these limitations, we propose an Accelerated Column Generation (ACG) framework. This framework incorporates a defect-aware interval selection heuristic that significantly improves convergence speed. As a result, the proposed approach is particularly well-suited for real-world industrial settings, where mother plates exhibit diverse dimensions, defects overlap heavily, and quality constraints are tightly enforced.

The main contributions of this work are summarized as follows:

1) We consider a two-dimensional cutting problem, in which small items with quality grades have to be cut from large objects with surface defects such that the total revenue of a steel plant is maximized. 2D_G_MHLOPP_DQ is formulated as a mixed-integer programming (MIP) model via introducing a width permutation scheme and an allocation intervals generation algorithm.
2) We develop a column generation-based (CG-based) algorithm that embeds the width permutation scheme. A pricing subproblem is partitioned by width permutations and converted into several dimension-reduction ones.
3) We present an accelerated CG (ACG) algorithm to improve the performance of CG-based algorithm. In the pricing process, three heuristic accelerating strategies are adopted to find a subset of subproblems in each iteration process and a subset of elements in each subproblem. The numerical results demonstrate the effectiveness of the algorithm in real-life scenarios.

The remainder of this work is organized into five sections. Section II defines 2D_G_MHLOPP_DQ precisely. Sections III and IV introduce CG-based algorithm and ACG algorithm in detail, respectively. Section V reports computational results obtained from a series of instances, which are randomly generated based on the actual cutting process. Section VI concludes with a summary and an outlook on future research.

II. PROBLEM STATEMENT AND MATHEMATICAL MODEL

2D_G_MHLOPP_DQ can be stated as follows: a set of large objects, each of which has a fixed width and length. These large objects have rectangular defects with different sizes, grades, and positions, and any two defects can overlap. For each small item type, its width, length, weight, value, and maximum acceptable quality grade are known. The objective is to maximize the total revenue by cutting defective large objects with two-staged guillotine cuts.

Before introducing the MIP model, we first describe a width permutation scheme and an allocation intervals algorithm. The former can handle two-staged guillotine cutting patterns, which is adapted from the width combination scheme of Zhang *et al*. [19]. The latter can deal with rectangular defects, which is inspired by Fasano *et al*. [20]. Both of them are beneficial to model the concerned problem.

A. Width Permutation Scheme

In a two-staged guillotine cutting pattern, the horizontal 1-cuts produce shelves. After that, small items are obtained with the vertical 2-cuts. In order to enumerate the positions of 1-cuts for defective large objects, we devise a width permutation scheme with the following notations.

Set:

I Index set of small items;
J Index set of large objects;
P Index set of width permutations;
S Index set of shelves;
$\boldsymbol{w}$ Set of small items' widths;
$\mathcal{W}$ Set of large objects' widths.

Parameters:

W_{spj} Width of shelf s in width permutation p for large object j, where $s \in S$, $p \in P$, $j \in J$.

Example 1: To better understand the working mechanism of the width permutation scheme, we consider a small instance with $|I| = 4$ and $|J| = 1$, and report the results in Table II.

Based on the given small items' widths, we obtain five width permutations for a large object that contains four defects of three grades. As shown in Fig. 2, the red lines represent 1-cuts' positions, each of which determines the width of a shelf. It is worth noting that horizontal 1-cuts can affect the number of defects. See Fig. 2(d),defect 2 of grade 5 is split into two segments due to the horizontal cut, which requires renumbering and separate treatment in the cutting process. For the same set of input data, we obtain four width combinations [19] for the large object without defects. The reason for the different results is that symmetry should be considered in this work. See Figs. 2(b) and (c), defect 2 is cut into different shelves, and we need to consider both cases.

TABLE II Results of width permutation scheme

Input	$w = \{1000, 1500, 2000, 1000\}$	
	$\mathcal{W} = \{3000\}$	
Output	$\|P\| = 5$	$W_{111} = 3000$, $\|S\| = 1$
		$W_{121} = 1000$, $W_{221} = 2000$, $\|S\| = 2$
		$W_{131} = 2000$, $W_{231} = 1000$, $\|S\| = 2$
		$W_{141} = 1500$, $W_{241} = 1500$, $\|S\| = 2$
		$W_{151} = 1000$, $W_{251} = 1000$, $W_{351} = 1000$, $\|S\| = 3$

Algorithm 1 Allocation Intervals Generation (AIG)

Input: $l, \mathcal{G}, \mathcal{L}, \mathcal{M}, \mathcal{P}$
Output: $\mathcal{A}$
Initialize $\mathcal{A} \leftarrow \emptyset$, $k \leftarrow 0$, $I_k \leftarrow \emptyset$
Execute width permutation scheme to obtain p
for $j = 1$ to $|J|$ **do**
  for $p = 1$ to $|P|$ **do**
    for $s = 1$ to $|S|$ **do**
      $b_{kspj} \leftarrow 0$, $e_{kspj} \leftarrow L_j$
      $\mathcal{A} \leftarrow \mathcal{A} \cup [b_{kspj}, e_{kspj}]$, $k \leftarrow k + 1$
      for $i = 1$ to $|I|$ **do**
        if $m_i = \max\{m_i\}$ **then**
          $I_k \leftarrow \{i\} \cup I_k$
        end if
      end for
      for $d = 1$ to $|D|$ **do**
        if $l^b_{dj} \geq l^{\min}$ **then**
          $b_{kspj} \leftarrow 0$, $e_{kspj} \leftarrow l^b_{dj}$
          $\mathcal{A} \leftarrow \mathcal{A} \cup [b_{kspj}, e_{kspj}]$, $k \leftarrow k + 1$
        end if
        if $L_j - l^e_{dj} \geq l^{\min}$ **then**
          $b_{kspj} \leftarrow l^e_{dj}$, $e_{kspj} \leftarrow L_j$
          $\mathcal{A} \leftarrow \mathcal{A} \cup [b_{kspj}, e_{kspj}]$, $k \leftarrow k + 1$
        end if
        for $i = 1$ to $|I|$ **do**
          if $g_d \leq m_i$ **then**
            $I_k \leftarrow \{i\} \cup I_k$
          end if
        end for
      end for
    end for
  end for
end for
return $\mathcal{A}$

B. Allocation Intervals Algorithm

In the cutting process, the method of dealing with defects is the key to solving 2D_G_MHLOPP_DQ. Since sizes, grades, and positions of defects are given, we do not need to recompute them during the solution process. For each small item type,

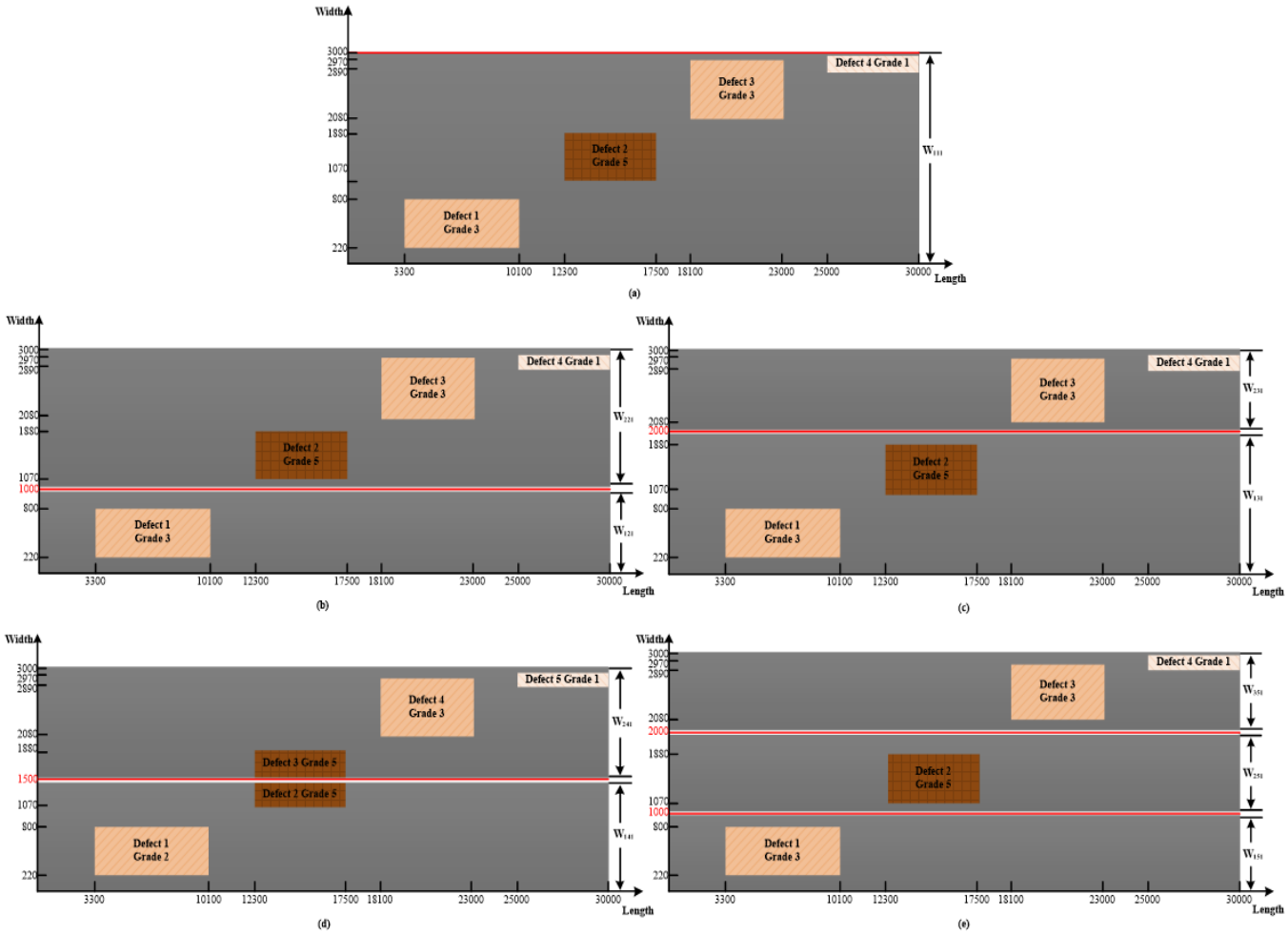


Fig. 2. Details of 1-cuts' positions.

allocation intervals are known after the widths of shelves are given. Any small item of the same type must be completely placed in one of the corresponding allocation intervals of each shelf. In order to enumerate all allocation intervals, we develop an allocation intervals generation algorithm with the following notations.

Set:

D	Index set of defects;
K	Index set of allocation intervals;
I_k	Index set of small items used in allocation interval k, $k \in K$, i.e., $I_k \subseteq I$;
$\boldsymbol{l}$	Set of small items' lengths;
$\boldsymbol{p}$	Set of width permutations;
$\mathcal{A}$	Set of allocation intervals;
$\mathcal{G}$	Set of defect grades;
$\mathcal{L}$	Set of large objects' lengths;
$\mathcal{M}$	Set of maximum acceptable quality grades of small items;
$\mathcal{P}$	Set of positions of defects.

Parameters:

b_{kspj}	Beginning position of allocation interval k on shelf s in width permutation p for large object j, $k \in K$, $s \in S$, $p \in P$, $j \in J$.
e_{kspj}	Ending position of allocation interval k on shelf s in width permutation p for large object j, $k \in K$, $s \in S$, $p \in P$, $j \in J$.
l_i	Length of small item i, $i \in I$.
l^b_{dspj}	Beginning position of defect d on shelf s in width permutation p along the lengthwise of large object j, $d \in D$, $s \in S$, $p \in P$, $j \in J$.
l^e_{dspj}	Ending position of defect d on shelf s in width permutation p along the lengthwise of large object j, $d \in D$, $s \in S$, $p \in P$, $j \in J$.
w^b_{dspj}	Beginning position of defect d on shelf s in width permutation p along the widthwise of large object j, $d \in D$, $s \in S$, $p \in P$, $j \in J$.
w^e_{dspj}	Ending position of defect d on shelf s in width permutation p along the widthwise of large object j, $d \in D$, $s \in S$, $p \in P$, $j \in J$.

The pseudocode of allocation intervals generation algorithm is given in Algorithm 1. For simplicity in notations, we use the

$$
\begin{aligned}
&[w^b_{dpsj}, w^e_{dpsj}] \times [l^b_{dpsj}, l^e_{dpsj}] \\
&= \left\{ (x, y) \in \mathbb{R}^2 \;\middle|\; w^b_{dpsj} \le x \le w^e_{dpsj},\ l^b_{dpsj} \le y \le l^e_{dpsj}, \right. \\
&\qquad \left. d \in D, \quad s \in S, \quad p \in P, \quad j \in J \right\}
\end{aligned} \tag{1}
$$

definition given in (1) to represent the positions of defects.

Example 2. To better understand the working mechanism of the allocation intervals generation algorithm, we consider a small instance with $|J| = 1$, $|I| = 4$, and $|D| = 4$, and report the results in Table III.

From Table III, we find that some allocation intervals intersect in pairs, and define these intervals as overlapping intervals. We consider a small instance with four allocation intervals (see Fig. 3). Let F denote the set of overlapping intervals. We obtain $\mathcal{F} = \{(1,2),\ (1,3),\ (1,4),\ (2,4),\ (3,4)\}$.

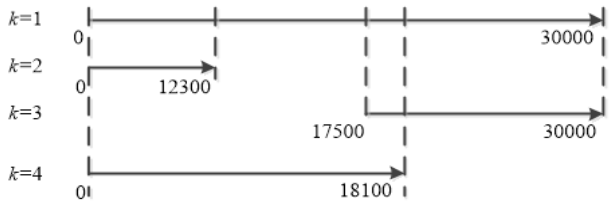


Fig. 3. Details of overlapping intervals.

C. MIP Model

The MIP model of 2D_G_MHLOPP_DQ is shown in (1)-(14) and uses the following additional notations.

Set:

K_f index set of overlapping intervals.

Parameters:

l_i length of small item i, $i \in I$.
w_i width of small item i, $i \in I$.
L_j length of large object j, $j \in J$.
W_j width of large object j, $j \in J$.
α_j penalty coefficient, which depends on the number of surface defects on large object j, $j \in J$.
β coefficient of large object.
$\underline{b}_{k_f spj}$ beginning position of overlapping interval k_f on shelf s of width permutation p of large object j, where $k_f \in K_f$, $s \in S$, $p \in P$, $j \in J$.
d_i demand of small item i, $i \in I$.
$\overline{e}_{k_f spj}$ ending position of overlapping interval k_f on shelf s of width permutation p of large object j, where $k_f \in K_f$, $s \in S$, $p \in P$, $j \in J$.
f_j production cost of large object j, $j \in J$.
w^e_i weight (in tons) of small item i, $i \in I$.
v^r_i revenue per ton of small item i, $i \in I$.
U_{ikspj} maximum number of small item i used in allocation interval k on shelf s of width permutation p of large object j, where $i \in I_k$, $k \in K$, $s \in S$, $p \in P$, $j \in J$, i.e.,

$$U_{ikspj} = \left\lfloor \frac{e_{kspj} - b_{kspj}}{l_i} \right\rfloor.$$

TABLE III Results of allocation intervals.

Input	$l = \{12300, 12800, 12000, 7500\}$	
	$L = \{30000\}$	
	$G = \{3, 5, 3, 1\}$	
	$M = \{2, 4, 2, 5\}$	
	$\lvert\mathcal{P}\rvert = 4$	$[220, 800] \times [3300, 10100]$
		$[1070, 1880] \times [12300, 17500]$
		$[2080, 2970] \times [18100, 23000]$
		$[2890, 3000] \times [25000, 30000]$
Output	$\lvert K\rvert = 26$	$b_{1111} = 0,\ e_{1111} = 30000,\ I_1 = \{4\}$
		$b_{2111} = 0,\ e_{2111} = 12300,\ I_2 = \{4\}$
		$b_{3111} = 17500,\ e_{3111} = 30000,\ I_3 = \{4\}$
		$b_{1121} = 0,\ e_{1121} = 30000,\ I_1 = \{2, 4\}$
		$b_{2121} = 10100,\ e_{2121} = 30000,\ I_2 = \{1, 2, 3, 4\}$
		$b_{1221} = 0,\ e_{1221} = 30000,\ I_1 = \{4\}$
		$b_{2221} = 0,\ e_{2221} = 12300,\ I_2 = \{4\}$
		$b_{3221} = 17500,\ e_{3221} = 30000,\ I_3 = \{4\}$
		$b_{1131} = 0,\ e_{1131} = 30000,\ I_1 = \{4\}$
		$b_{2131} = 0,\ e_{2131} = 12300,\ I_2 = \{4\}$
		$b_{3131} = 17500,\ e_{3131} = 30000,\ I_3 = \{4\}$
		$b_{1231} = 0,\ e_{1231} = 30000,\ I_1 = \{2, 4\}$
		$b_{2231} = 0,\ e_{2231} = 18100,\ I_2 = \{1, 2, 3, 4\}$
		$b_{1141} = 0,\ e_{1141} = 30000,\ I_1 = \{4\}$
		$b_{2141} = 0,\ e_{2141} = 12300,\ I_2 = \{4\}$
		$b_{3141} = 17500,\ e_{3141} = 30000,\ I_3 = \{4\}$
		$b_{1241} = 0,\ e_{1241} = 30000,\ I_1 = \{4\}$
		$b_{2241} = 0,\ e_{2241} = 12300,\ I_2 = \{4\}$
		$b_{3241} = 17500,\ e_{3241} = 30000,\ I_3 = \{4\}$
		$b_{1151} = 0,\ e_{1151} = 30000,\ I_1 = \{2, 4\}$
		$b_{2151} = 10100,\ e_{2151} = 30000,\ I_2 = \{1, 2, 3, 4\}$
		$b_{1251} = 0,\ e_{1251} = 30000,\ I_1 = \{4\}$
		$b_{2251} = 0,\ e_{2251} = 12300,\ I_2 = \{1, 3, 4\}$
		$b_{3251} = 17500,\ e_{3251} = 30000,\ I_3 = \{4\}$
		$b_{1351} = 0,\ e_{1351} = 30000,\ I_1 = \{2, 4\}$
		$b_{2351} = 0,\ e_{2351} = 18100,\ I_2 = \{1, 2, 3, 4\}$

Decision Variables:

n_{ikspj} number of small item i used in allocation interval k on shelf s of width permutation p of large object j, where $i \in I_k$, $k \in K$, $s \in S$, $p \in P$, $j \in J$.
u_{ikspj} equals 1 if small item i is placed in allocation interval k on shelf s of width permutation p of large object j, and 0 otherwise; where $i \in I_k$, $p \in P$, $s \in S$, $c \in C$, $j \in J$.
x_{kspj} equals 1 if allocation interval k is used on shelf s of width permutation p of large object j, and 0 otherwise; where $k \in K$, $s \in S$, $p \in P$, $j \in J$.
y_{kspj} equals 1 if allocation interval k is the longest one on shelf s of width permutation p of large object j, and 0 otherwise; where $k \in K$, $s \in S$, $p \in P$, $j \in J$.
z_{pj} equals 1 if width permutation p is selected by large object j, and 0 otherwise; where $p \in P$, $j \in J$.

The objective function (2) is to maximize the total revenue of a steel plant. Its first item represents revenue for placing small items on defective large objects, the second one is production cost and the third one is revenue for utilizing surplus materials from large objects.

$$\text{Maximize} \quad \sum_{j\in J}\sum_{p\in P}\sum_{s\in S}\sum_{k\in K}\sum_{i\in I_k}\left(n_{ikspj}w_i^e v_i^r - \alpha_j f_j\right) + \beta\left(W_j L_j - n_{ikspj} w_i l_i\right) \tag{2}$$

We have the following constraints. Constraints (3) ensure that exactly one width permutation is selected for each large object, i.e., $\forall j \in J$.

$$\sum_{p\in P} z_{pj} = 1, \quad \forall j \in J \tag{3}$$

Constraints (4) state that the width of a shelf in a width permutation for a large object is determined by the widest small item in this shelf, i.e., $\forall i \in I_k,\ k \in K,\ s \in S,\ p \in P,\ j \in J$.

$$u_{ikspj} \cdot w_i \le W_{spj} \cdot z_{pj} \tag{4}$$

Constraints (5) limit the number of small items in an allocation interval on a shelf in a width permutation for a large object, i.e., $\forall i \in I_k,\ k \in K,\ s \in S,\ p \in P,\ j \in J$.

$$n_{ikspj} \le u_{ikspj} \cdot U_{ikspj} \tag{5}$$

Constraints (6) impose that the total number of each small item type should not exceed its demand, i.e., $\forall i \in I_k$.

$$\sum_{j\in J}\sum_{p\in P}\sum_{s\in S}\sum_{k\in K} n_{ikspj} \le d_i \tag{6}$$

Constraints (7) are association constraints between decision variables, $\forall k \in K, s \in S, p \in P, j \in J$.

$$x_{kspj} \le \sum_{i\in I_k} u_{ikspj}, \tag{7}$$

Constraints (8) ensure that the sum of small items' lengths cannot exceed the length of an allocation interval on a shelf in a width permutation for a large object, i.e., $\forall k \in K, s \in S, p \in P, j \in J$.

$$\sum_{i\in I_k} n_{ikspj} l_i \le (e_{kspj} - b_{kspj})\, x_{kspj} \tag{8}$$

Likewise, constraints (9) show that the sum of small items' lengths cannot exceed the length of an overlapping interval on a shelf in a width permutation for a large object, i.e., , $\forall s \in S, p \in P, j \in J$.

$$\sum_{k_f\in K_f}\sum_{i\in I_k} n_{ik_fspj} l_i \le \bar{e}_{k_fspj} - \underline{b}_{k_fspj} + \left(2 - \sum_{k_f\in K_f} x_{k_fspj}\right) L_j \tag{9}$$

Similarly, constraints (10) require that the sum of small items' lengths cannot exceed the length of the longest interval on a shelf in a width permutation for a large object, i.e., $\forall s \in S, p \in P, j \in J$.

$$\sum_{k\in K}\sum_{i\in I_k} n_{ikspj} l_i \le (e_{kspj} - b_{kspj}) y_{kspj} + (1 - y_{kspj}) L_j \tag{10}$$

Constraints (11)-(12) are association constraints between decision variables: $\sum_{k\in K} y_{kspj} = 0$ only if $z_{pj} = 0$, and that $y_{kspj} = 0$ if $x_{kspj} = 0, \forall k \in K, s \in S, p \in P, j \in J$.

$$\sum_{k\in K} y_{kspj} = z_{pj} \tag{11}$$

$$y_{kspj} \le x_{kspj} \tag{12}$$

Constraints (13) guarantee that allocation interval k is the longest interval on a shelf in a width permutation for a large object only if $x_{kspj} = y_{kspj} = 1$, i.e., $\forall k \in K \setminus \{|K|\}, s \in S, p \in P, j \in J$.

$$\sum_{\substack{k'\in K \\ k'>k}} x_{k'spj} \le (2 - x_{kspj} - y_{kspj}) L_j \tag{13}$$

Constraints (14)–(15) define the type of variables, i.e., $\forall i \in I_k, k \in K, s \in S, p \in P, j \in J$,

$$n_{ikspj} \in \mathbb{Z}_+, \tag{14}$$

$$u_{ikspj},\ x_{kspj},\ y_{kspj},\ z_{pj} \in \{0, 1\}. \tag{15}$$

Since constraints (9) depend on the value of parameters $\underline{b}_{kspj}$ and $\bar{e}_{kspj}$, several cases should be considered.

Case 1: If $b_{kspj} = b_{k'spj},\ e_{kspj} < e_{k'spj}$,

Subcase 1a: If $l^{\min} \le e_{k'spj} - e_{kspj}$, then

$$\underline{b}_{kspj} = b_{kspj} = b_{k'spj}, \quad \bar{e}_{kspj} = e_{k'spj}.$$

Subcase 1b: If $l^{\min} > e_{k'spj} - e_{kspj}$, then

$$\underline{b}_{kspj} = b_{kspj} = b_{k'spj}, \quad \bar{e}_{kspj} = e_{kspj}.$$

Case 2: If $e_{kspj} = e_{k'spj}, b_{kspj} < b_{k'spj}$, the corresponding subcases can be dealt with as in Case 1.

Case 3: If $b_{kspj} < b_{k'spj}, e_{kspj} < e_{k'spj}$,

Subcase 3a: If $l^{\min} \le e_{k'spj} - e_{kspj}, l^{\min} \le b_{k'spj} - b_{kspj}$, then

$$\underline{b}_{kspj} = b_{kspj}, \quad \bar{e}_{kspj} = e_{k'spj}.$$

Subcase 3b: If $l^{\min} \le e_{k'spj} - e_{kspj}, l^{\min} > b_{k'spj} - b_{kspj}$, then

$$\underline{b}_{kspj} = b_{k'spj}, \quad \bar{e}_{kspj} = e_{k'spj}.$$

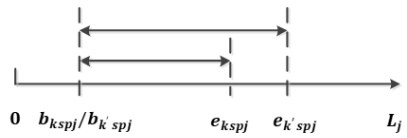
Subcase 3c: If $l^{\min} > e_{k'spj} - e_{kspj}, l^{\min} \le b_{k'spj} - b_{kspj}$, then

$$\underline{b}_{kspj} = b_{kspj}, \quad \bar{e}_{kspj} = e_{kspj}.$$

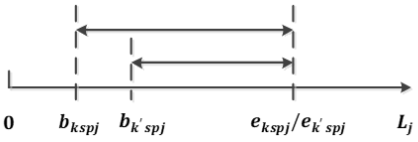
Subcase 3d: If $l^{\min} > e_{k'spj} - e_{kspj}, l^{\min} > b_{k'spj} - b_{kspj}$, then

$$\underline{b}_{kspj} = b_{k'spj}, \quad \bar{e}_{kspj} = e_{kspj}.$$

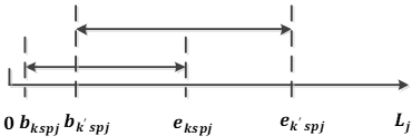
Case 4: If $b_{kspj} > b_{k'spj}, e_{kspj} < e_{k'spj}$, the corresponding subcases can be dealt with in the same way as in Case 3.



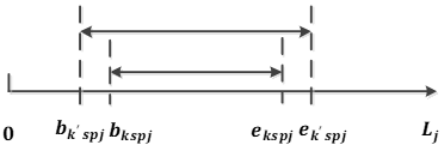
(a) Overlapping intervals for Case 1.



(b) Overlapping intervals for Case 2.



(c) Overlapping intervals for Case 3.



(d) Overlapping intervals for Case 4.

Fig. 4. Overlapping intervals for four cases.

III. COLUMN GENERATION ALGORITHM

The MIP model in Section II is capable of solving small-sized instances (containing a smaller number of item types) in a reasonable time. However, its computational efficiency is expected to degrade as the instance size increases. To overcome this difficulty, we propose a CG-based algorithm.

As it is usual in CG algorithm, the full problem is called the master problem (MP), while the linear program with only a subset of columns is called the restricted master problem (RMP). RMP and the pricing subproblem (PP) are solved iteratively, where the former passes to the latter the dual variables in order to find promising columns (here cutting patterns), i.e., having positive reduced costs. If no column of positive reduced cost can be found then the algorithm is terminated. Otherwise, one or more columns with positive reduced costs are added to RMP to improve the current solution, and the algorithm iterates. Fig. 5 illustrates the solving process of 2D_MHLOPP_DQ via using CG-based algorithm, in which each PP is partitioned by a width permutation scheme.

A. Master Problem

Each decision variable of MP represents the decision on selecting a two-staged two-dimensional cutting pattern for a large object with surface defects. The MP model is described in (15)-(18) with the following additions.

Sets:

C Index set of all possible cutting patterns.

Parameters:

v_{cj} Value of cutting pattern c of large object $j, \forall c \in C, j \in J$;

a_{icj} Number of times small item i occurs in cutting pattern c of large object $j, \forall i \in I, c \in C, j \in J$.

Decision Variables:

z_{cj} Equals 1 if large object j selects cutting pattern c; 0 otherwise, $\forall c \in C, j \in J$.

Objective function:

$$\text{Maximize} \quad \sum_{j\in J}\sum_{c\in C} v_{cj} z_{cj} + \sum_{j\in J} \beta W_j L_j - \alpha f \tag{16}$$

Subject to:

$$\sum_{j\in J}\sum_{c\in C} a_{icj} z_{cj} \le d_i, \quad \forall i \in I \tag{17}$$

$$\sum_{c\in C} z_{cj} = 1, \quad \forall j \in J \tag{18}$$

$$z_{cj} \in \{0,1\}, \quad \forall c \in C, j \in J \tag{19}$$

The objective function (16) maximizes the total revenue of a steel plant. Constraints (17) ensure that the demand of each small item type is not exceeded. Constraints (18) impose that exactly one cutting pattern is selected for each large object. Constraints (19) define the valid values for the decision variables.

CG-based algorithm iterates between the Restricted Master Problem (RMP) and the Pricing Problem (PP), where the RMP is similar to the original master problem (MP), but considers only a subset of cutting patterns $C' \subseteq C$. To find the optimal solution of the linear program, we first relax the integrality constraints (19) as $0 \le z_c^j \le 1$. Then, the linear relaxation of the RMP is solved to obtain a set of dual variable values, which are used to construct the reduced costs of potential new columns. The PP is formulated with the reduced cost as its objective function, and aims to improve the objective value of the MP by generating feasible two-staged, two-dimensional cutting patterns with positive reduced costs.

B. Pricing Subproblem

We adopt a width permutation scheme to convert a two-dimensional cutting pattern into several cutting patterns with width-fixed shelves. The scheme is successfully applied to decrease the dimension of a subproblem according to the study of [4]. PPs mentioned hereafter are all partitioned by a width permutation scheme, so that a height permutation corresponds to a PP.

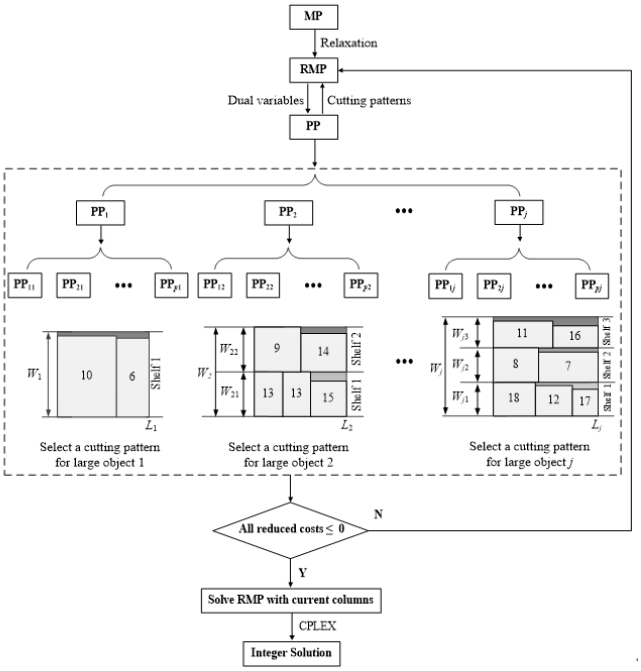


Fig. 5. Solving process of 2D_MHLOPP_DQ.

Since large objects with surface defects are considered as homogeneous, without confusion involved, the indexes p ($p \in P$) and j ($j \in J$) are omitted from the variables in PP_{pj}. We introduce the variables n_{iks}, u_{iks}, x_{ks}, and y_{ks} which correspond to the variables $n_{ikspj}, u_{ikspj}, x_{kspj}$, and y_{kspj} in the original problem. PP_{pj} can be formulated as the following model:

Objective function:

$$\text{Maximize} \quad \sum_{s\in S}\left(\sum_{k\in K}\sum_{i\in I_k} n_{iks}\left(w_i^e v_i^r - \beta w_i l_i - \pi_i\right) - \mu\right) \tag{20}$$

Subject to:

$$u_{iks} w_i \le W_s, \quad \forall i \in I_k, k \in K, s \in S \tag{21}$$

$$n_{iks} \le u_{iks} U_{iks}, \quad \forall i \in I_k, k \in K, s \in S \tag{22}$$

$$\sum_{s\in S}\sum_{k\in K} n_{iks} \le d_i, \quad \forall i \in I \tag{23}$$

$$x_{ks} \le \sum_{i\in I_k} u_{iks}, \quad \forall k \in K, s \in S \tag{24}$$

$$\sum_{i\in I_k} n_{iks} l_i \le (e_{ks} - b_{ks}) x_{ks}, \quad \forall k \in K, s \in S \tag{25}$$

$$\sum_{k_f\in K_f}\sum_{i\in I_k} n_{ik_fs} l_i \le \bar{e}_{k_fs} - \underline{b}_{k_fs} + \left(2 - \sum_{k_f\in K_f} x_{k_fs}\right) L, \quad \forall s \in S \tag{26}$$

$$\sum_{k\in K}\sum_{i\in I_k} n_{iks} l_i \le (e_{ks} - b_{ks}) y_{ks} + (1 - y_{ks}) L, \quad \forall s \in S \tag{27}$$

$$\sum_{k\in K} y_{ks} = 1, \quad \forall s \in S \tag{28}$$

$$y_{ks} \le x_{ks}, \quad \forall k \in K, s \in S \tag{29}$$

$$\sum_{\substack{k'>k\\ k'\in K}} x_{k's} \le (2 - x_{ks} - y_{ks}) L, \quad \forall s \in S \tag{30}$$

$$n_{iks} \in \mathbb{Z}_+, \quad \forall i \in I_k, k \in K, s \in S \tag{31}$$

$$u_{iks}, x_{ks}, y_{is} \in \{0, 1\}, \quad \forall i \in I_k, k \in K, s \in S \tag{32}$$

The objective function (20) maximizes the reduced cost of a PP, where π_i ($\pi_i \le 0, i \in I$) and μ are the dual variables associated with constraints (17) and (18), respectively. Constraints (21)–(32) correspond to constraints (4)–(15), which together formulate the feasibility of a cutting pattern with width-fixed shelves.

At each time that PP is solved, a cutting pattern c of large object j with associated coefficients v_c^j and a_{ic}^j is generated and provided to MP. These two values can be calculated as:

$$\begin{aligned} v_c^j &= \sum_{s\in S}\sum_{k\in K}\sum_{i\in I_k} n_{iks}\left(w_i^e v_i^r - \beta w_i l_i\right), \quad \forall c \in C, j \in J \\ a_{ic}^j &= \sum_{s\in S}\sum_{k\in K}\sum_{i\in I_k} n_{iks}, \quad \forall c \in C, j \in J \end{aligned} \tag{33}$$

For a given height permutation of a large object, we solve the corresponding PP to generate a new column based on the dual variables' value, where each column represents a feasible two-staged two-dimensional cutting pattern. All columns with positive reduced costs are added to MP in each iteration. The CG-based algorithm is terminated when no such cutting pattern can be found.

C. CG Procedure

The CG-based algorithm must start with a feasible solution. This can be achieved using a greedy heuristic, which generates a set of initial columns for the linear relaxation of the RMP. For each large object, we introduce $\mathbf{a}^j$ as the set of columns $\mathbf{a}_c^j = \left(a_{1c}^j, \ldots, a_{|I|c}^j\right)^\top$ and $\mathbf{v}^j$ as the set of corresponding values v_c^j. The greedy heuristic for generating an initial solution proceeds as follows, where d denotes the demand set of small items, $\mathbf{v}^r$ denotes the revenue set of small items, and $\mathbf{w}^e$ denotes the weight set of small items.

Let rc_j denote a reduced cost and a_j denote a column, which can be obtained by solving PP_j, ($j \in J$).The framework of the CG-based algorithm is presented in Algorithm 3.

Algorithm 2 Initial solution (IS)

Input: $d, l, \mathbf{v}^r, w, \mathbf{w}^e, \mathcal{M}, \mathcal{L}, \mathcal{W}$
Output: $\mathbf{a}, \mathbf{v}$

Initialize $\mathbf{a} \leftarrow \emptyset, \quad \mathbf{v} \leftarrow \emptyset, \quad I^o \leftarrow \emptyset$
for $j = 1$ to $|J|$ **do**
  for $i = 1$ to $|I|$ **do**
    if $m_i = 5$ **then**
      $a_{i0}^j \leftarrow \min\left\{d_i, \left\lfloor \frac{W_j}{h_i} \right\rfloor \times \left\lfloor \frac{L_j}{l_i} \right\rfloor\right\}$
      $v_0^j \leftarrow \left\lfloor \frac{W_j}{h_i} \right\rfloor \times \left\lfloor \frac{L_j}{l_i} \right\rfloor \times \left(w_i^e \times v_i^r - \beta \times w_i \times l_i\right)$
    end if
  end for
end for
return $\mathbf{a}, \mathbf{v}$

IV. ACCELERATED COLUMN GENERATION ALGORITHM

In practice, the number of cutting patterns is very large. Let C denote the set of all feasible cutting patterns, and $C \subseteq C$ represent the subset of cutting patterns included in the Restricted Master Problem (RMP). The cardinality $|C|$ indicates the number of cutting patterns currently considered in the RMP. The direct application of CG is not effective, since $|C|$ increases as the number of iterations grows. Numerical results reveal that efficiency can be greatly improved by the following heuristic acceleration strategies.

A. Heuristic Acceleration Strategies

We utilize problem-specific domain knowledge to explore the efficient width permutations, small items, and allocation intervals, which can potentially speed up the convergence of the process substantially. Heuristic acceleration strategies of efficient width permutation (EWP) and efficient small items (ESI) have been successfully applied to accelerate the CG-based algorithm. We refer to the study of [4] for details. In this work, we focus on describing the heuristic acceleration strategy of efficient interval allocations. In order to estimate which allocation intervals are efficient, we first design the following evaluation function based on the cost performance of small item i, where ζ is a given constant. Next, we sort the corresponding allocation intervals in descending order according to the value of δ_{ik} and select the first η allocation intervals for each pricing subproblem, where $0 < \eta \leq |K|$, $\eta \in \mathbb{Z}^+$. These selected efficient allocation intervals belong to a subset of $\mathcal{A}$, which is denoted as $\mathcal{A}'$ ($\mathcal{A}' \subseteq \mathcal{A}$). Note that given a set A, we use to denote a subset of A'.

$$\delta_{ikspj} = \begin{cases} \dfrac{e_{kspj} - b_{kspj}}{l_i}, & \zeta < \dfrac{e_{kspj} - b_{kspj}}{l_i} < 1 \\ 0, & \text{otherwise} \end{cases} \qquad (34)$$
$$i \in I_k, \quad k \in K, \quad s \in S, \quad p \in P, \quad j \in J$$

Equation (34) defines the efficiency score δ_{ikspj}, which measures the suitability of allocating a small item i to a specific allocation interval $[b_{kspj}, e_{kspj}]$. The score is calculated as the ratio between the interval length and the length of item i, subject to a lower threshold ζ. Only intervals where this ratio satisfies $\zeta < \frac{e_{kspj} - b_{kspj}}{l_i} < 1$ are considered effective. This evaluation helps to filter out inefficient intervals and select the most promising allocation intervals, thereby improving the computational efficiency of the column generation algorithm.

Algorithm 3 CG-based algorithm

Input: $d, l, w, \mathbf{w}^e, \mathbf{v}^r, \mathcal{G}, \mathcal{L}, \mathcal{M}, \mathcal{P}, \mathcal{W}$
Output: an integer solution

Execute AIG($l, \mathcal{G}, \mathcal{L}, \mathcal{M}, \mathcal{P}$) to obtain $\mathcal{A}$
Execute IS($d, l, \mathbf{v}^r, w, \mathbf{w}^e, L, W$) to obtain $\mathbf{A}, \mathbf{v}$
repeat
  Solve linear relaxation of RMP with $\mathbf{A}, \mathbf{v}$ to obtain values of dual variables π
  stop $\leftarrow$ true
  for $j = 1$ to $|J|$ **do**
    Solve PP_j with parameters
    $d, l, p, w, \mathbf{w}^e, \mathbf{v}^r, \mathcal{A}, \mathcal{L}, \mathcal{W}$ to obtain rc_j, a_j, v_j
    if $rc_j < 0$ **then**
      Add a_j to $\mathbf{A}$
      Add v_j to $\mathbf{v}$
      stop $\leftarrow$ false
      break
    end if
  end for
until stop = true
Solve RMP with current columns to optimality using CPLEX
return an integer solution

B. ACG Procedure

The procedure for efficient allocation intervals is presented in Algorithm 4 as follows. According to the heuristic acceleration strategies of EWP and ESI, we introduce p' as the set of efficient width permutations and I_k' as the set of efficient small items. The framework of the ACG algorithm is presented in Algorithm 5. These three heuristic strategies complement each other in the ACG framework. EWP focuses on selecting width permutations with high utilization potential, ESI filters effective small items through interval evaluation to balance computational efficiency and solution quality, while EAI further eliminates inefficient allocation intervals based on area utilization. Together, they form a progressive filtering mechanism that significantly reduces the solution space while maintaining solution quality.

V. COMPUTATIONAL EXPERIMENTS

A. Experimental Setup

To evaluate the effectiveness and scalability of the proposed algorithms, we conduct a series of computational experiments on problem instances of varying sizes. All algorithms are implemented in C++ and executed on a personal desktop equipped with an Intel Core i5 processor (4 cores, 3.3 GHz), 4 GB of RAM, and running Windows 7 64-bit. The commercial solver CPLEX 12.6 (with default settings) is used to solve the master problems and linearized subproblems.

Algorithm 4 Efficient Allocation Intervals (EAI)

Input: $l, \mathcal{G}, \mathcal{L}, \mathcal{M}, \mathcal{P}, \zeta, \eta$
Output: $\mathcal{A}'$

Execute AIG$(l, \mathcal{G}, \mathcal{L}, \mathcal{M}, \mathcal{P})$ to obtain $\mathcal{A}$
for $j = 1$ to $|J|$ **do**
  for $p = 1$ to $|\mathcal{P}|$ **do**
    for $s = 1$ to $|\mathcal{S}|$ **do**
      for $k = 1$ to $|\mathcal{K}|$ **do**
        for $i = 1$ to $|I_k|$ **do**
          Compute δ_{ikspj}
        end for
      end for
    end for
  end for
end for
Sort δ_{ikspj} in descending order, select the first η corresponding allocation intervals
return $\mathcal{A}'$

Algorithm 5 ACG algorithm

Input: $d, l, w, \mathbf{w}^e, \mathbf{v}^r, \mathcal{G}, \mathcal{L}, \mathcal{M}, \mathcal{P}, \mathcal{W}, \gamma, \zeta, \eta, \tau$
Output: an integer solution

Execute AIG$(l, \mathcal{G}, \mathcal{L}, \mathcal{M}, \mathcal{P})$ to obtain $\mathcal{A}$
Execute IS$(d, l, \mathbf{v}^r, w, \mathbf{w}^e, \mathcal{L}, \mathcal{W})$ to obtain $\mathcal{A}, \mathbf{v}$
do
Solve linear relaxation of RMP with $\mathcal{A}, \mathbf{v}$ and obtain values of dual variables π
stop $\leftarrow$ true
Execute EWP to obtain p' and select the first γ corresponding width permutations
Execute ESI to obtain I'_k and select the first τ corresponding small items
Execute EAI$(l, \mathcal{G}, \mathcal{L}, \mathcal{M}, \mathcal{P}, \zeta, \eta)$ to obtain $\mathcal{A}'$ and select the first η corresponding allocation intervals
for $j = 1$ to $|J|$ **do**
  Solve PP_j with $d', l', p', \mathbf{v}^{r'}, w', \mathbf{w}^{e'}, \mathcal{A}', \mathcal{L}, \mathcal{W}$ to obtain rc'_j, a'_j, v'_j
  if $rc'_j < 0$ **then**
    Add a'_j to $\mathcal{A}$
    Add v'_j to $\mathbf{v}$
    stop $\leftarrow$ false
    break
  end if
end for
while stop $\neq$ true
Solve RMP with current columns to optimality by using CPLEX
return an integer solution

We first adopt the MIP model as a baseline for performance comparison. As shown in Table V, MIP fails to solve more than half of the 60 test instances within the specified time limit, particularly for medium- and large-sized problems. This result highlights the limited scalability of the MIP approach due to the high computational complexity of the 2D_G_MHLOPP_DQ problem.

TABLE IV Parameters and their values

Parameter	Value
α	{1.1, 1.2, 1.3, 1.4}
β	$0.01 \times 7.85 \times 3000 \times 10^{-6}$
f	30500
$\lvert\mathcal{I}\rvert$	{100, 200, 300, 400, 500, 600, 700, 800, 900, 1000}
$\mathcal{G}$	{1, 2, 3, 4, 5}
$\mathcal{L}$	{5200, 4500, 4000, 3500, 3000}
$\mathcal{M}$	{0, 1, 2, 3, 4, 5}
$\mathcal{W}$	{50000, 45000, 40000, 35000, 30000}

In contrast, the CG algorithm demonstrates strong capability in producing high-quality solutions within acceptable runtimes for small-sized instances. Given the observed limitations of MIP, the subsequent experiments focus on comparing the CG algorithm with the proposed ACG algorithm.

We systematically evaluate CG and ACG across a wide range of instance sizes, reporting on three key performance metrics: optimality gaps, average runtime, and scalability. This comparative analysis provides insights into the practical applicability of the proposed methods for real-world steel cutting optimization scenarios.

B. Data and Problem Instances

To ensure the practical relevance of our computational experiments, we collect production data from a real-world steel plant and use it to determine the value ranges for key parameters in our model. Table IV summarizes the parameter settings, which reflect typical dimensions, quality grades, and production costs observed in industrial operations. Based on these parameter settings, we generate 10 groups of test instances, each corresponding to a different number of small item types $\mathcal{I}$. Within each group, four defect configurations are defined according to the number of surface defects on the mother plates: 2–4, 5–7, 8–10, and 11–13 defects, respectively. For each defect configuration, five random instances are generated, resulting in a total of $10 \times 4 \times 5 = 200$ test instances. A time limit of 3600 seconds is imposed on each algorithm iteration.

To simulate realistic surface defect conditions, we generate defect dimensions (width and height) and severity levels based on statistical distributions derived from plant data. Specifically, defect widths are sampled from a uniform distribution $U(10, \text{mm}, 50, \text{mm})$, and defect heights from $U(10, \text{mm}, 100, \text{mm})$, reflecting typical variation in physical defect sizes. Defect severity levels $\mathcal{G} = 1, 2, 3, 4, 5$ follow an empirically derived distribution based on operational experience: 10% at level 1, 20% at level 2, 30% at level 3, 25% at level 4, and 15% at level 5. This setting ensures that the test instances closely approximate the variability and defect profile found in real production environments.

To highlight the performance difference between the baseline MIP model and the CG-based algorithm, we summarize key experimental results in Table V. The results show that the MIP model struggles with scalability: for larger instances (e.g., Group S3), it fails to return any solutions within the time limit, while the CG-based approach consistently solves all cases with significantly lower runtime.

TABLE V Performance difference between MIP and CG-based algorithm.

Group	$\|P\|$	$\|S\|$	#instances		Ave. time (s)	
			MIP	CG-based	MIP	CG-based
S1	66	2	16	20	949	175
S2	143	3	13	20	1680	425
S3	218	3	0	20	3600	953

TABLE VI Computational results for the medium-sized instances.

Group	$\|P\|$	$\|K\|$	$\|S\|$	Parameters of ACG			Ave. gap (%)	Ave. time (s)	
				γ	τ	η		CG-based	ACG
M1	219	579	3	88	190	307	1.15	724	375
M2	229	633	3	94	240	347	2.22	1609	491
M3	259	755	3	109	290	439	4.92	2562	695

Note: Gap = $\frac{\text{Obj}^{\text{ACG}} - \text{Obj}^{\text{CG-based}}}{\text{Obj}^{\text{CG-based}}}$.

C. Computational Results for Small-sized Instances

We first compare CG-based algorithm with MIP, using the small-sized instances. The computational results are reported in Table V. For each instance group, we introduce four performance measures:

- $|P|$: the average number of width permutations;
- $|S|$: the average number of shelves;
- #instances: the number of instances for which MIP and CG-based algorithm terminated successfully within the given time;
- Ave. time: the average computational time in seconds required to solve instances in each group.

From Table V, it can be taken that the direct usage of MIP is testified to be limited. For 31 out of the 60 instances, the given time is reached for solving MIP by CPLEX. It should be noted that CG-based algorithm is capable of generating high-quality solutions for all small-sized instances within a reasonable time. Thus, in the next section, we focus on our evaluation of the performances of CG-based algorithm and ACG algorithm.

D. Computational Results for Medium-sized Instances

We next implement CG-based algorithm and ACG algotihm to solve medium-sized instances. The computational results are reported in Table Table VI. For each instance group, we introduce additional five performance measures:

- $|K|$: the average number of allocation intervals;
- γ: the average number of efficient width permutations;
- τ: the average number of efficient small items;
- η: the average number of efficient allocation intervals;
- Ave. gap: the average gap between objective value associated with an integer solution of CG-based algorithm and objective value corresponding to an integer solution of ACG algorithm.

In order to obtain the best parameter combination of ACG algorithm, we adopt a fixed parameter method. For each group, parameters are fixed one by one, and their values in the five random instances are the same.

TABLE VII Performance of ACG algorithm for the large-sized instances.

Group	$\|P\|$	$\|K\|$	$\|S\|$	Parameters of ACG			Ave. gap (%)	Ave. time (s)
				γ	τ	η		
L1	266	725	3	107	350	442	3.54	899
L2	287	818	3	118	390	507	5.79	973
L3	318	949	3	134	450	598	6.23	1768
L4	292	893	3	126	490	572	7.58	2250

Note: Gap = $\frac{\text{UB} - \text{Obj}^{\text{ACG}}}{\text{UB}}$.

Several remarks could be obtained from the results in Table VI. First, we select fewer width permutations, small items, and allocation intervals in each iteration of ACG algorithm. It is showed that three heuristic acceleration strategies are efficient. As one may observe, 41.11% of the total width permutations, 47.93% of the total small items, and 55.33% of the total allocation intervals have a remarkable effect on the improvement of objective value in each iterative solution process. Second, both CG-based algorithm and ACG algorithm could find high-quality solutions for medium-sized instances. Although two algorithms could deal with these instances, ACG significantly outperformed CG-based algorithm in terms of solution quality and computing times.

E. Computational Results for Large-sized Instances

We finally evaluate the performance of ACG algorithm, using the large-sized instances. The computational results are reported in Table VII, where UB denotes the optimal objective value of linear relaxation MP when ACG algorithm stops.

Several remarks could be obtained similar to those in subsection C. First, ACG algorithm could find high-quality solutions for large-sized instances. Second, 41.658% of total width permutations, 49.44% of total small items, and 62.53% of total allocation intervals have a remarkable effect on the improvement of objective value in each iterative solution process. It is showed that three heuristic acceleration strategies are still efficient as the type of small items increases. In summary, our experimental results show that ACG algorithm is a computationally efficient algorithm for 2D_MHLOPP_DQ.

F. Limitations and Future Works

This work demonstrates the practical value of the proposed 2D_G_MHLOPP_DQ model and the effectiveness of the ACG algorithm. However, several limitations remain that warrant further investigation.

First, the current model assumes a static production environment and does not explicitly account for dynamic factors such as order priority changes, machine breakdowns, or unplanned interruptions. In real-world steel manufacturing, these factors can significantly affect production scheduling and layout decisions, potentially rendering static solutions suboptimal under changing conditions [21]–[24].

Second, although the ACG algorithm improves computational efficiency and performs well on large instances, its scalability

and robustness may be challenged in ultra-large-scale and highly heterogeneous scenarios. Enhancing algorithmic adaptability and performance under such conditions remains an open research problem.

Looking ahead, we plan to extend the 2D_G_MHLOPP_DQ model by incorporating steel grade compatibility between mother plates and customer orders. This addition will further increase the model's complexity, as it introduces new constraints on item placement and order satisfaction [25]–[29].

In addition, future work will explore the integration of advanced mathematical techniques, including effective cutting-plane inequalities [30] and bounding strategies [31], to improve solution quality and reduce computation time. These enhancements are expected to facilitate the development of exact or near-exact methods capable of solving large-scale industrial problems with higher precision and reliability [32]–[34].

VI. CONCLUSIONS

This work investigates a two-dimensional cutting optimization problem encountered in steel manufacturing, where surface defects and quality grade constraints significantly influence cutting layout decisions. To address the problem's computational challenges, we propose an ACG algorithm that builds upon the traditional CG framework by embedding three heuristic acceleration strategies: efficient width permutations, efficient small item selection, and efficient allocation interval pruning.

Extensive numerical experiments confirm that both CG and ACG algorithms are capable of producing high-quality solutions for small- and medium-sized instances. For large-scale and highly heterogeneous problems, the ACG algorithm consistently outperforms the standard CG approach in terms of runtime and solution quality. These findings highlight the strong potential of the proposed method for supporting practical decision-making in real-world steel production environments, improving material utilization, and reducing production waste.

Nevertheless, this study has several limitations. The current model assumes a static environment and does not consider dynamic production factors such as order priority changes or equipment failures. Additionally, while ACG performs well on large instances, further improvements in scalability and robustness are needed for ultra-large-scale industrial applications.

Future work will extend the model to incorporate additional industrial factors such as steel grade compatibility between mother plates and customer orders. We also plan to explore the integration of advanced techniques, including valid inequalities and bounding methods, to further enhance algorithmic efficiency and solution precision for real-time industrial deployment.

REFERENCES

[1] C. Carnieri, G. A. Mendoza, and W. G. Luppold, "Optimal cutting of dimension parts from lumber with a defect: a heuristic solution procedure," *Forest Products Journal*, vol. 43, no. 9, pp. 66–72, 1993.

[2] V. Neidlein, A. C. G. Vianna, M. N. Arenales, and G. Wäscher, "Two-dimensional guillotineable-layout cutting problems with a single defect—an and/or-graph approach," in *Operations Research Proceedings Part 3*. Berlin-Heidelberg: Springer, 2009, pp. 85–90.

[3] M. Afsharian, A. Niknejad, and G. Wäscher, "A heuristic dynamic programming based approach for a two-dimensional cutting problem with defects," *OR Spectrum*, vol. 36, no. 4, pp. 971–999, 2014.

[4] B. Durak and D. T. Aksu, "Dynamic programming and mixed integer programming based algorithms for the online glass cutting problem with defects and production targets," *International Journal of Production Research*, vol. 55, no. 9, pp. 1–14, 2017.

[5] M. Martin, P. H. Hokama, R. Morabito, and P. Munari, "The constrained two-dimensional guillotine cutting problem with defects: an ilp formulation, a benders decomposition and a cp-based algorithm," *International Journal of Production Research*, vol. 58, no. 58, pp. 2712–2729, 2020.

[6] P. C. Gilmore and R. E. Gomory, "Multistage cutting stock problems of two and more dimensions," *Operations Research*, vol. 13, no. 1, pp. 94–120, 1965.

[7] L. Ozdamar, "The cutting-wrapping problem in the textile industry: optimal overlap of fabric lengths and defects for maximizing return based on quality," *International Journal of Production Research*, vol. 38, no. 6, pp. 1287–1309, 2000.

[8] R. Ghodsi and F. Sassani, "Real-time optimum sequencing of wood cutting process," *International Journal of Production Research*, vol. 43, no. 6, pp. 1127–1141, 2005.

[9] R. Aboudi and P. Barcia, "Determining cutting stock patterns when defects are present," *Annals of Operations Research*, vol. 82, no. 1, pp. 343–354, 1998.

[10] F. Parreño, R. Alvarez-Valdes, and J. F. Oliveira, "Beam search algorithm for three-stage rectangular cutting patterns," *Computers & Operations Research*, vol. 124, p. 105067, 2020.

[11] A. Martin, O. Schneider, and R. Müller, "Benders decomposition for multi-motherplate cutting with overlapping defects," *European Journal of Operational Research*, vol. 294, no. 2, pp. 601–614, 2021.

[12] H. Durak and D. T. Aksu, "Online dynamic programming for defective material cutting with point defects," *Journal of Manufacturing Systems*, vol. 65, pp. 382–395, 2022.

[13] W. Chen, Q. Li, and H. Zhang, "Deep reinforcement learning for dynamic defect handling in cutting processes," in *Proceedings of the IEEE International Conference on Industrial Informatics (INDIN)*, 2023, pp. 1–6.

[14] T. Wang and Y. Zhang, "Heuristic algorithm for multi-stage cutting with regional defects," *International Journal of Production Research*, vol. 60, no. 15, pp. 4782–4797, 2022.

[15] E. Silva *et al.*, "Irregular cutting with defect avoidance via column generation," *Computers & Operations Research*, vol. 148, p. 106024, 2022.

[16] B. Moreira *et al.*, "Quality-constrained column generation for hardwood cutting," *European Journal of Operational Research*, vol. 306, no. 1, pp. 78–91, 2023.

[17] L. Caro *et al.*, "Complexity analysis for cutting problems with overlapping defects," *Journal of Global Optimization*, vol. 81, pp. 723–745, 2021.

[18] F. Furini *et al.*, "Column generation for cutting and packing: Recent advances," *EURO Journal on Computational Optimization*, vol. 11, p. 100060, 2023.

[19] Q. Zhang, S. X. Liu, R. Y. Zhang, and S. J. Qin, "Column generation algorithms for mother plate design in steel plants," *OR Spectrum*, vol. 43, no. 1, pp. 127–153, 2020.

[20] G. Fasano and J. D. Pintér, Eds., *Optimized Packings with Applications*. Springer, 2015, vol. 105.

[21] X. W. Guo, F. Guo, L. Qi, J. Wang, S. X. Liu, S. Qin, and W. T. Wang, "Modeling and optimization of multi-product human-robot collaborative hybrid disassembly line balancing with resource sharing," *IEEE Transactions on Computational Social Systems*, 2025.

[22] X. W. Guo, L. Chen, L. Qi, J. Wang, S. Qin, M. Chatterjee, and Q. Kang, "Multi-factory disassembly process optimization considering worker posture," *IEEE Transactions on Computational Social Systems*, 2025.

[23] T. T. Wei, X. W. Guo, M. Zhou, J. Wang, S. X. Liu, S. Qin, and Y. Tang, "A multi-objective discrete harmony search optimizer for disassembly line balancing problems considering human factors," *IEEE Transactions on Human-Machine Systems*, vol. 55, no. 2, pp. 124–133, April 2025.

[24] L. Qi, L. Liang, W. J. Luan, T. Lu, X. W. Guo, and Q. T. A. Talukder, "Integrated control strategies for freeway bottlenecks with vehicle platooning," *International Journal of Artificial Intelligence and Green Manufacturing*, vol. 1, no. 1, pp. 46–56, April 2025.

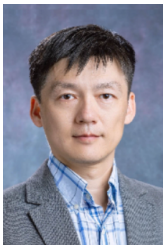
[25] Y. Ren, C. Zhang, F. Zhao, G. Tian, W. Lin, L. Meng, and H. Li, "Disassembly line balancing problem using interdependent weights-based multi-criteria decision making and 2-optimal algorithm," *Journal of Cleaner Production*, vol. 174, pp. 1475–1486, 2018.
[26] J. Liu, Z. Zhou, D. T. Pham, W. Xu, C. Ji, and Q. Liu, "Collaborative optimization of robotic disassembly sequence planning and robotic disassembly line balancing problem using improved discrete bees algorithm in remanufacturing," *Robotics and Computer-Integrated Manufacturing*, vol. 61, p. 101829, 2020.
[27] L. Liuke, Z. Zeqiang, Z. Binsen, and C. Ning, "Optimization of multi-objective disassembly line balancing problem using immune mechanism cooperative genetic algorithm," *Information and Control*, vol. 47, pp. 671–679, 2018.
[28] L. Zhu, Z. Zhang, and Y. Wang, "A pareto firefly algorithm for multi-objective disassembly line balancing problems with hazard evaluation," *International Journal of Production Research*, vol. 56, pp. 7354–7374, 2018.
[29] G. Alp and A. F. Alkaya, "Hyperheuristic based migrating birds optimization algorithm for a fairness oriented shift scheduling problem," *Mathematical Problems in Engineering*, vol. 2021, p. 6756588, 2021.
[30] Z. Zikai, T. Qiuhua, L. Zixiang, and H. Dayong, "An efficient migrating birds optimization algorithm with idle time reduction for type-i multi-manned assembly line balancing problem," *Journal of Systems Engineering and Electronics*, vol. 32, pp. 286–296, 2021.
[31] Z. Li, M. N. Janardhanan, A. S. Ashour, and N. Dey, "Mathematical models and migrating birds optimization for robotic u-shaped assembly line balancing problem," *Neural Computing and Applications*, vol. 31, pp. 9095–9111, 2019.
[32] Y. J. Ji, Z. Y. Zhao, S. X. Liu, and X. Y. Yong, "Machine learning-based prediction of surplus material in intelligent production processes," *International Journal of Artificial Intelligence and Green Manufacturing*, vol. 1, no. 1, pp. 25–34, April 2025.
[33] S. Yan, H. F. Guo, and S. X. Liu, "A partition stacking classification framework with oversampling for quality prediction of aluminum alloy ingots," *International Journal of Artificial Intelligence and Green Manufacturing*, vol. 1, no. 1, pp. 35–45, April 2025.
[34] S. J. Qin, F. Guo, J. Wang, L. Qi, J. X. Wang, X. W. Guo, and Z. Y. Zhao, "Expanded discrete migratory bird optimizer for circular disassembly line balancing with tool deterioration and replacement," *International Journal of Artificial Intelligence and Green Manufacturing*, vol. 1, no. 1, pp. 14–24, 2025.

Qi Zhang received her B.S. degree and M.S. degree in applied mathematics from Shenyang Normal University, Shenyang, China in 2012 and 2015, respectively, and Ph.D. degree in Systems Engineering from Northeastern University, Shenyang, China in 2022. She is currently a lecturer of the College of Information Engineering at Shenyang University of Chemical Technology. Her research focuses on bin packing problem, steel production scheduling and planning, mathematical programming, and intelligent optimization algorithm.

Yang Xing received his B.E. degree in n Electronic Science and Technology from Shenyang University of Chemical Technology, shenyang, China, in 2021. She is currently a Graduate student of the Information Engineering at Shenyang Chemical of University, Shenyang, China. Her research interests include smart manufacturing and algorithm optimization.

Changtian Zhang received his B.E. degree in automation from Jilin Institute of Architectural Science and Technology, Changchun, China in 2023. He is currently a graduate student majoring in control engineering at Shenyang University of Chemical Technology. His research interests include ceramic production scheduling and planning, intelligent optimization algorithms, and smart manufacturing systems.

Xiaoxu Sun received her B.E. in Artiffcial Intelligence from Shenyang University, China, in 2024. Currently, she is a graduate student in the school of Information Engineering at Shenyang University of Chemical Technology. Her research interests include steel production scheduling and planning and intelligent optimization algorithms.



Bin Hu received the B.S. degree from Xi'an Jiaotong University, Xi'an, Shaanxi, China, in 2000, the double M.S. degrees from Xi'an Jiaotong University, and Monmouth University, Long Branch, NJ, USA, in 2005 and 2018, respectively, and the Ph.D. degree from Rutgers University, New Brunswick, NJ, in 2023. From 2005 to 2016, he was an Assistant Professor with Xi'an University of Posts and Telecommunications, Xi'an. He is currently an Assistant Professor with the Department of Computer Science and Technology, Kean University. His research interests include mobile computing and sensing, cybersecurity and privacy, and efficient deep learning.

Arup Das is a leading expert in Artificial Intelligence and Machine Learning with extensive experience in applied AI, generative AI, and data-driven innovation across industries. He has held senior roles at UiPath, Avenue One, Compass, and Machine Analytics, where he led the development of AI-driven systems for sectors such as financial services, healthcare, and manufacturing. He holds advanced degrees from Cornell University, Villanova University, and Stony Brook University, and is co-author of The Generative AI Practitioner's Guide, reflecting his commitment to bridging industry and academia through applied innovation.