Integra: Journal of Integrated Mathematics and Computer Science

Journal Homepage: https://integrajimcs.com/index.php/integra/index

integra
Journal of Integrated Mathematics and Computer Science

Research Paper

Approximation Method for Solving Rectangular Guillotine Cutting-Stock Problems

Abdul Rahman[1], Ihda Hasbiyati[1], Wamiliana[2], Moh Danil Hendry Gamal[1]*

[1] *Department of Mathematics, Faculty of Mathematics and Natural Sciences, Universitas Riau, Pekanbaru, 28293, Indonesia*
[2] *Department of Mathematics, Faculty of Mathematics and Natural Sciences, Universitas Lampung, Bandar Lampung, 35145, Indonesia*
***Corresponding author**: mdh.gamal@lecturer.unri.ac.id

Keywords
Rectangular Cutting-Stock Problem, Guillotine Cut, Column Generation, Dynamic Programming, Approximation Method, Cutting Pattern, Combinatorial Optimization

Abstract
Rectangular cutting-stock problems are fundamental optimization challenges in manufacturing industries, where the objective is to cut rectangular items from larger sheets while minimizing material waste. This paper introduces an approximation approach to solve rectangular cutting-stock problems with guillotine cuts, where cutting patterns are produced through a column generation framework. In the proposed approach, the master problem determines an optimal combination of cutting patterns, while new patterns are generated through pricing subproblems formulated as knapsack problems. Dynamic programming is employed to efficiently solve these knapsack subproblems, enabling the identification of high-quality cutting patterns. By iteratively enriching the pattern set, the method approximates the optimal solution without requiring exhaustive enumeration of all feasible patterns. The method offers a practical and scalable solution for rectangular cutting-stock problems encountered in real-world practice.

Received: 27 December 2025, Accepted: 12 March 2026
https://doi.org/10.26554/integra.20263148

1. INTRODUCTION

Rectangular cutting-stock problem (RCSP) consists of cutting a set of smaller rectangular items from larger rectangular stock sheets while satisfying demand and minimizing waste or the number of stock sheets used. This problem arises in many industrial contexts such as metal cutting, glass manufacturing, wood processing, textile production, and paper industries. The RCSP is known to be NP-hard, and its computational difficulty increases significantly when additional constraints, such as guillotine cutting requirements, are imposed [1, 2, 3, 4, 5, 6, 7].

In many practical cutting environments, guillotine cuts are mandatory due to machine limitations or operational simplicity. A guillotine cut is a straight cut that spans the entire length or width of a rectangle, partitioning it into two smaller rectangles. Guillotine cutting structures limit the range of feasible cutting patterns, but they enable solutions to be applied efficiently in real-world practice [1]. Consequently, the rectangular guillotine cutting-stock problem has attracted significant attention in the literature [2, 7, 8, 9].

A major milestone in cutting-stock optimization was achieved by Gilmore and Gomory [10], who modelled the one-dimensional cutting-stock problem as a linear programming formulation containing an exponential number of variables, each corresponding to a cutting pattern. They introduced the column generation approach, where the model initially considers only a limited subset of patterns, and additional patterns are generated iteratively by solving a pricing subproblem. This approach has since become the foundation for solving large-scale cutting-stock and packing problems [11, 12].

The extension of column generation to rectangular cutting-stock problems presents additional challenges, particularly in pattern generation. In the two-dimensional case, the pricing subproblem often takes the form of a knapsack-type or packing problem, which itself is NP-hard. Dynamic programming has been widely used to address this difficulty, as it provides an effective way to generate high-quality cutting patterns un-

der guillotine constraints [1, 13]. The integration of dynamic programming into the column generation framework allows efficient exploration of the pattern space while maintaining computational tractability.

Several researchers have proposed column generation approaches for two-dimensional guillotine cutting problems. Beasley [1] developed one of the earliest exact algorithms for two-dimensional guillotine cutting, while Zhang et al. [2] analyzed different mathematical formulations extending the Gilmore–Gomory model to two dimensions. Later, Cintra et al. [13] integrated dynamic programming with column generation to address rectangular cutting-stock and strip-packing problems, showing the effectiveness of this hybrid approach.

More recent studies have focused on heuristic and approximation methods to handle large-scale industrial instances. Furini et al. [14] introduced a column generation–based heuristic for the two-stage rectangular guillotine cutting-stock problem with multiple stock sizes, demonstrating that near-optimal solutions can be obtained efficiently. Similar approaches using modified column generation techniques have been reported in [15, 16] where guillotine constraints are explicitly incorporated into the pattern generation process. Heuristic enhancements and staged cutting structures have also been explored to further improve computational performance [4, 17, 18, 19].

Despite these advances, solving large rectangular guillotine cutting-stock problems to optimality remains computationally demanding. Therefore, approximation methods that combine column generation with dynamic programming represent a promising compromise between solution quality and computational effort. Such methods exploit the strengths of column generation in handling large variable spaces and the efficiency of dynamic programming in generating feasible guillotine patterns [12, 13, 14].

In this study, an approximation method is put forward for solving the rectangular cutting-stock problem with guillotine cuts by integrating column generation and dynamic programming. The main contributions of this study include, first, a mathematical formulation based on a restricted master problem for guillotine cutting patterns, along with an efficient dynamic programming algorithm to solve the pricing subproblem.

2. PROBLEM FORMULATION

In general, cutting-stock is carried out by industries that produce materials in standard sizes, which are then cut according to the requested order sizes. The process of cutting a rectangular stock with standard length and width into multiple smaller pieces is referred to as the rectangular cutting-stock problem.

Assume that a steel plate company manufactures plates measured in feet (ft), each with a length of 12 ft and a width of 10 ft. The company receives orders for steel plates of various sizes. The company must fulfill these demands by cutting the produced steel plates of size 12 ft × 10 ft according to the orders. It is assumed that the number of steel plates produced is unlimited. Table 1 shows an example of the orders.

Table 1. Example of Orders

Order (i)	Ordered Size (Length (ft) × Width (ft))	Quantity Ordered (Sheets)
midrule 1	7 × 3	18
2	3 × 4	22
3	4 × 5	14

In practice, to fulfill an order, the cutting blades are adjusted to cut standard sizes according to a pattern. Figure 1 shows three types of cutting patterns that may be applied to an order.

Although other cutting patterns exist, for the purposes of this example the discussion is limited to Patterns A, B, and C as shown in Figure 1. Several patterns can be combined to fulfill orders with sizes of 7 ft × 3 ft, 3 ft × 4 ft, and 4 ft × 5 ft. The following are two examples of feasible combinations that can be used:

(i) Combination 1: Cut 6 sheets of the standard size 12 ft × 10 ft using Pattern A and 6 sheets using Pattern C.
(ii) Combination 2: Cut 11 sheets of the standard size 12 ft × 10 ft using Pattern B and 7 sheets using Pattern C.

To determine which pattern combination is better, it can be evaluated by considering the cutting waste. The following is the cutting waste produced by each combination:

(i) Combination 1:: $6 \times 12 + 6 \times 17 = 174$ ft^2.
(ii) Combination 2: : $11 \times 33 + 7 \times 17 = 482$ ft^2.

Furthermore, any surplus (excess) production with dimensions of 7 ft × 3 ft, 3 ft × 4 ft, and 4 ft × 5 ft must be considered in the calculations as cutting waste. In Combination 1, Pattern A produces 24 sheets of size 3 ft × 4 ft and 18 sheets of size 4 ft × 5 ft, while Pattern C produces 12 sheets of size 4 ft × 5 ft and 18 sheets of size 7 ft × 3 ft. Thus, in Combination 1 there are surplus productions for the 3 ft × 4 ft size (surface area = 12 ft^2) as many as 2 sheets, equal to 2×12 ft^2 $= 24$ ft^2, and for the 4 ft × 5 ft size (surface area = 20 ft^2) as many as 16 sheets, equal to 16×20 ft^2 $= 320$ ft^2. The total surplus in Combination 1 is therefore $24 + 320 = 344$ ft^2.

In Combination 2, Pattern B produces 22 sheets of size 3 ft × 4 ft and 33 sheets of size 7 ft × 3 ft, while Pattern C produces 14 sheets of size 4 ft × 5 ft and 21 sheets of size 7 ft × 3 ft. Thus, in Combination 2 there is surplus production for the 7 ft × 3 ft size (surface area = 21 ft^2) of 35 sheets, equal to 35×21 ft^2 $= 735$ ft^2. So,

total cutting waste for Combination 1 = $174 + 344 = 518$ ft^2 and

total cutting waste for Combination 2 = $482 + 735 = 1217$ ft^2.

Therefore, Combination 1 is better than Combination 2 because it produces less cutting waste. Furthermore, to determine the optimal solution to the problem, the first step is to identify all possible patterns. Let P denotes a steel plate with dimensions 3 ft × 4 ft, Q a steel plate with dimensions 4 ft × 5 ft, and R a steel plate with dimensions 7 ft × 3 ft. Table 2 illustrates the possible

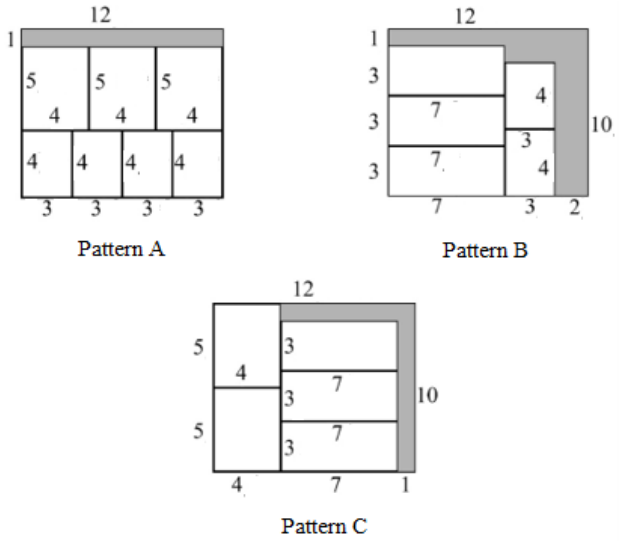


Figure 1. Three Possible Cutting Patterns

cutting patterns obtained by cutting the standard steel plate size of 12 ft × 10ft.

From Table 2, there are thirteen possible cutting patterns suitable with the purpose of this paper, that is patterns 7, 9, 10, 11, 20, 21, 24, 27, 28, 29, 31, 32, and 33.

Let x_j be the number of standard stock size 12 ft × 10 ft cut according to pattern j where $j = 1, 2, \ldots, 34$. The linear programming formulation for the above cutting-stock problem is then given as follows: The objective function can be written as

$$\min z = x_1 + x_2 + x_3 + \cdots + x_{34}.$$

This means that the cutting waste can be minimized by minimizing the number of standard stocks used. Then there are three constraints to be considered, that is

(i) Constraint 1: at least 18 sheets of size 7ft × 3ft should be produced.
(ii) Constraint 2: at least 22 sheets of size 3ft × 4ft should be produced.
(iii) Constraint 3: at least 14 sheets of size 4ft × 5ft should be produced.

Since the total number of size 7ft × 3ft cut is given by $5x_1 + 4x_2 + 4x_3 + 3x_4 + 3x_5 + 3x_6 + 3x_7 + 3x_8 + 2x_9 + 2x_{10} + 2x_{11} + 2x_{12} + 2x_{13} + 2x_{14} + 2x_{15} + x_{16} + x_{17} + x_{18} + x_{19} + x_{20} + x_{21}$, then Constraint 1 becomes $5x_1 + 4x_2 + 4x_3 + 3x_4 + 3x_5 + 3x_6 + 3x_7 + 3x_8 + 2x_9 + 2x_{10} + 2x_{11} + 2x_{12} + 2x_{13} + 2x_{14} + 2x_{15} + x_{16} + x_{17} + x_{18} + x_{19} + x_{20} + x_{21} \geq 18$.

Using similar way, Constraint 2 can be written as $x_1 + 3x_2 + x_3 + 3x_4 + 2x_5 + x_6 + x_7 + 5x_9 + 4x_{10} + 4x_{11} + 3x_{12} + 2x_{13} + x_{14} + 7x_{16} + 6x_{17} + 4x_{18} + 3x_{19} + x_{20} + 10x_{22} + 9x_{23} + 8x_{24} + 8x_{25} + 6x_{26} + 6x_{27} + 5x_{28} + 4x_{29} + 3x_{30} + 2x_{31} + x_{32} \geq 22$ and Constraint 3 as $x_6 + 2x_8 + x_{10} + x_{12} + 2x_{13} + 3x_{14} + 3x_{15} + x_{17} + 2x_{18} + 2x_{19} + 3x_{20} + 4x_{21} + x_{25} + 2x_{26} + x_{27} + 2x_{28} + 3x_{29} + 3x_{30} + 4x_{31} + 5x_{32} + 6x_{33} + 4x_{34} \geq 14$.

Model (1) below shows the the linear programming model for the above cutting-stock problem which is described before.

$$\left.\begin{aligned}
&\min\ z = x_1 + x_2 + x_3 + \cdots + x_{31} + x_{34} \\
&\text{subject to} \\
&\quad 5x_1 + 4x_2 + 4x_3 + 3x_4 + 3x_5 + 3x_6 + 3x_7 \\
&\qquad + 3x_8 + 2x_9 + 2x_{10} + 2x_{11} + 2x_{12} + 2x_{13} \\
&\qquad + 2x_{14} + 2x_{15} + x_{16} + x_{17} + x_{18} + x_{19} + x_{20} \\
&\qquad + x_{21} \geq 18, \\
&\quad x_1 + 3x_2 + x_3 + 3x_4 + 2x_5 + x_6 + x_7 + 5x_9 \\
&\qquad + 4x_{10} + 4x_{11} + 3x_{12} + 2x_{13} + x_{14} + 7x_{16} \\
&\qquad + 6x_{17} + 4x_{18} + 3x_{19} + x_{20} + 10x_{22} + 9x_{23} \\
&\qquad + 8x_{24} + 8x_{25} + 6x_{26} + 6x_{27} + 5x_{28} + 4x_{29} \\
&\qquad + 3x_{30} + 2x_{31} + x_{32} \geq 22, \\
&\quad x_6 + 2x_8 + x_{10} + x_{12} + 2x_{13} + 3x_{14} + 3x_{15} \\
&\qquad + x_{17} + 2x_{18} + 2x_{19} + 3x_{20} + 4x_{21} + x_{25} \\
&\qquad + 2x_{26} + x_{27} + 2x_{28} + 3x_{29} + 3x_{30} + 4x_{31} \\
&\qquad + 5x_{32} + 6x_{33} + 4x_{34} \geq 14, \\
&\quad x_1, x_2, x_3, \ldots, x_{34} \geq 0 \text{ and integer}
\end{aligned}\right\} \quad (1)$$

In general, let L denote the standard stock length, W denote the standard stock width, n represent the total number of feasible cutting patterns, and x_j represent the number of standard stock pieces cut using pattern j, l_i := the number of sheets with length i demanded where $l_i \leq L$, w_i := the number of sheets with width i demanded where $w_i \leq W$, a_{ij} := the number of sheets with length l_i and width w_i of pattern j and b_i := the number of sheets of size $l_i \times w_i$ where $i = 1, \ldots, m$. Equation (2) shows the general form of the cutting-stock problem, which seeks to minimize material waste during cutting, is given as follows:

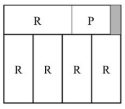
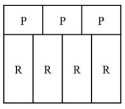
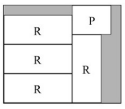
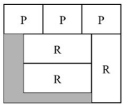
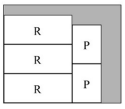
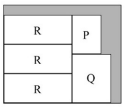
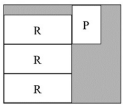
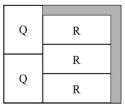
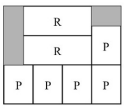
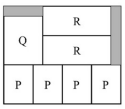
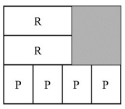
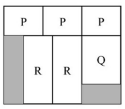
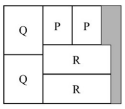
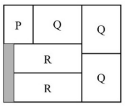
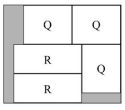
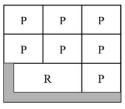
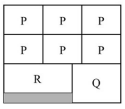
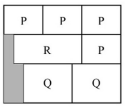
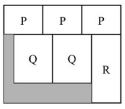
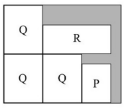
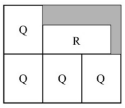
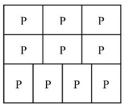
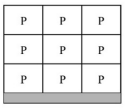
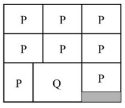
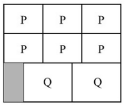
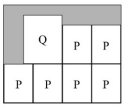
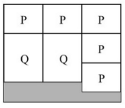
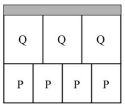
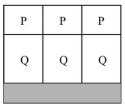
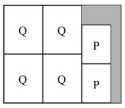
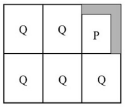
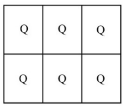
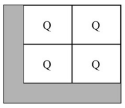
$$\begin{aligned}
&\min\ z = x_1 + x_2 + \cdots + x_n, \\
&\text{subject to}\quad a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n \geq b_i, \quad i = 1, 2, \ldots, m, \\
&\qquad x_j \geq 0 \text{ and integer}, \quad j = 1, 2, \ldots, n.
\end{aligned} \quad (2)$$

Based on this background, this paper examines methods for solving the rectangular cutting-stock problem under the assumption that cutting is carried out using a single type of stock with predetermined length and width. The stock supply is assumed to be unlimited, and the required pieces are not allowed to be rotated. A column generation approach is applied to derive knapsack problem formulations, which are then solved using dynamic programming techniques. The procedure begins by solving the knapsack problem with a length constraint, followed by the knapsack problem with a width constraint. Each knapsack formulation is developed individually according to the specific case.

3. COLUMN GENERATION TECHNIQUE

Column generation is a technique in linear programming used to solve very large optimization problems that have too many

Table 2. Figures of all possible cutting patterns

Pattern j	Figure	Pattern j	Figure	Pattern j	Figure	Pattern j	Figure
1.		2.		3.		4.	
5.		6.		7.		8.	
9.		10.		11.		12.	
13.		14.		15.		16.	
17.		18.		19.		20.	
21.		22.		23.		24.	
25.		26.		27.		28.	
29.		30.		31.		32.	
33.		34.					

variables (columns or patterns in the cutting-stock problems) to handle all at once. Instead of considering every possible variable (pattern) from the start, column generation adds variables (patterns) only when they are needed.

Equation (2) can be expressed in matrix form and written in its standard form as follows.

$$\begin{aligned} \min z &= c_B^T x_B + c_N^T x_N, \\ \text{subject to } & Bx_B + Nx_N = b, \\ & x_B, x_N \geq 0 \text{ and integer.} \end{aligned}$$

where x_B and x_N denote the basic and nonbasic variables,

respectively, and B and N represent the column matrices corresponding to the variables x_B and x_N, respectively. In current iteration of simplex method, let the associated basis be $B = (A_1, \dots, A_i, \dots, A_m)$ where A_i is a column vector with dimension m, $c_B^T = (c_1, c_2, \dots, c_m)$ be the coefficients of the objected function associated with $A_1, \dots, A_i, \dots, A_m$. Then, based on the linear programming, cutting pattern j is considered beneficial if its associated reduced cost

$$z_j - c_j = c_B^T B^{-1} A_j - c_j,$$

having positive value in minimization problems, where

$$A = (a_{1j}, a_{2j}, \dots, a_{mj})^T$$

is a vector that denotes the number of sheets with l_i and w_i being produced from cutting pattern j [20, 21].

For a minimization problem, a pattern is advantageous when it yields the maximum positive value of $z_j - c_j$. Thus, a column generation technique is used to identify promising columns, which is equivalent to solving a subproblem that shown in Equation (3).

$$\begin{aligned} \max\ & t = d_1 y_1 + d_2 y_2 + \cdots + d_m y_m - 1 \\ \text{subject to } & r_1 y_1 + r_2 y_2 + \cdots + r_m y_m \\ & \le R \\ & y_i \ge 0 \text{ and integer.} \end{aligned} \tag{3}$$

where d_i is an ith dual value of $c_B^T B^{-1}$, y_i is a nonnegative integer, R is the standard stock dimension, and r_i is the number of sheets of dimension i demanded with $r_i \le R$. Subproblem (3) is called knapsack problem.

4. RESULTS AND DISCUSSION

Look back at Equation (1). The initial basis is obtained from the pure cutting pattern. The pure cutting pattern produces 3 sheets of size 7 ft × 3 ft, 8 sheets of size 3 ft × 4 ft, and 6 sheets of size 4 ft × 5 ft. So, the initial basis for the sizes 7 ft × 3 ft, 3 ft × 4 ft and 4 ft × 5 ft are

$$B_0 = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 8 & 0 \\ 0 & 0 & 6 \end{bmatrix} \quad \text{and} \quad B_0^{-1} = \begin{bmatrix} \frac{1}{3} & 0 & 0 \\ 0 & \frac{1}{8} & 0 \\ 0 & 0 & \frac{1}{6} \end{bmatrix}.$$

Dual values are obtained from the multiplication below

$$c_B B_0^{-1} = [1 \quad 1 \quad 1] \begin{bmatrix} \frac{1}{3} & 0 & 0 \\ 0 & \frac{1}{8} & 0 \\ 0 & 0 & \frac{1}{6} \end{bmatrix} = \begin{bmatrix} \frac{1}{3} & \frac{1}{8} & \frac{1}{6} \end{bmatrix}.$$

For the new basis, a pattern denoted by y_1, y_2 and y_3, will be determined its $z_j - c_j$ as

$$c_B B_0^{-1} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} - 1 = \frac{1}{3} y_1 + \frac{1}{8} y_2 + \frac{1}{6} y_3 - 1.$$

where y_1, y_2 and y_3 are nonnegative integer not larger than 12 ft, y_1, y_2 and y_3, and pattern y_1, y_2 and y_3 must satisfy

$$7y_1 + 3y_2 + 4y_3 \ge 12,$$

$$y_i \ge 0 \text{ and integer.}$$

To obtain a promising pattern is by solving the equivalent knapsack problem, namely

$$\begin{aligned} \max\ & t = \frac{1}{3} y_1 + \frac{1}{8} y_2 + \frac{1}{6} y_3 - 1, \\ \text{subject to } & 7y_1 + 3y_2 + 4y_3 \le 12, \\ & y_i \ge 0 \text{ and integer.} \end{aligned} \tag{4}$$

Equation (4) represents a knapsack problem with a length constraint, which is solved using dynamic programming. The computation begins at stage ($i = 1$) and progresses forward to the final stage.

Case 1: $7y_1 \le 12$.

Applying the dynamic programming approach, the optimal solution to the constraint $7y_1 \le 12$ is obtained when $y_1 = 1$, resulting in $e_1 = t = \frac{1}{3}$.

Case 2: $7y_1 + 3y_2 \le 12$.

Using dynamic programming method, the optimal solution of the case $7y_1 + 3y_2 \le 12$ is $y_1 = 0$, $y_2 = 4$ with $e_2 = t = \frac{1}{2}$.

Case 3: $7y_1 + 3y_2 + 4y_3 \le 12$.

The dynamic programming method yields the optimal solution $y_1 = 1$, $y_2 = 0$, $y_3 = 1$ for the case $7y_1 + 3y_2 + 4y_3 \le 12$ with $e_3 = t = \frac{1}{2}$.

Next, the knapsack problem with a width constraint is solved using the objective function values obtained from the length-constrained knapsack problem, leading to the formulation displayed on Equation (5)

$$\begin{aligned} \max\ & t = \frac{1}{3} y_1 + \frac{1}{2} y_2 + \frac{1}{2} y_3 - 1, \\ \text{subject to } & 3y_1 + 4y_2 + 5y_3 \le 10, \\ & y_i \ge 0 \text{ and integer.} \end{aligned} \tag{5}$$

To solve Equation (5), the process starts from the initial stage ($i = 1$) and proceeds to the final stage or moving forward.

Case 1: $3y_1 \le 10$.

Using dynamic programming method, the optimal solution for the case $3y_1 \le 10$ is $y_1 = 3$ with $t = t_1 = 0$.

Case 2: $3y_1 + 4y_2 \le 10$.

The dynamic programming method produces the optimal solution $y_1 = 2$, $y_2 = 1$ for the case $3y_1 + 4y_2 \leq 10$ with $t = t_2 = \frac{1}{6}$.

Case 3: $3y_1 + 4y_2 + 5y_3 \leq 10$.

The dynamic programming method yields the optimal solution $y_1 = 2$, $y_2 = 1$, $y_3 = 0$ for the case $3y_1 + 4y_2 + 5y_3 \leq 10$ with $t = t_3 = \frac{1}{6}$.

From cases 1, 2, and 3 under the width constraint, the maximum objective function value t is found in case 2, where $t = t_3 = \frac{1}{6}$. Thus, the optimal solution of the knapsack problem in Equation (5) is the integer solution $y_1 = 2$, $y_2 = 1$ and $y_3 = 0$, yielding $t = \frac{1}{6}$. Since this width-constrained solution corresponds to the length-constrained knapsack solution, the updated values are $y_1 = 2 \times 1 = 2$, $y_2 = 1 \times 4 = 4$ and $y_3 = 0$.

This corresponds to pattern 11 and the variable x_{11}. Thus, the value of $z_{11} - c_{11}$ is 1/6. Since the value of $t = 1/6$, the current cutting pattern is not yet optimal. To find a more promising pattern, the obtained pattern is inserted into the basis, necessitating the creation of a new right-hand side and column.

$$\text{The new } x_{11} \text{ column} = B_0^{-1}\begin{bmatrix}2\\4\\0\end{bmatrix} = \begin{bmatrix}\frac{1}{3} & 0 & 0\\ 0 & \frac{1}{8} & 0\\ 0 & 0 & \frac{1}{6}\end{bmatrix}\begin{bmatrix}2\\4\\0\end{bmatrix} = \begin{bmatrix}\frac{2}{3}\\ \frac{1}{2}\\ 0\end{bmatrix}.$$

$$\text{The new right hand side} = B_0^{-1}b = \begin{bmatrix}\frac{1}{3} & 0 & 0\\ 0 & \frac{1}{8} & 0\\ 0 & 0 & \frac{1}{6}\end{bmatrix}\begin{bmatrix}18\\22\\14\end{bmatrix} = \begin{bmatrix}6\\ \frac{22}{8}\\ \frac{14}{6}\end{bmatrix}.$$

Ratio test shows that x_{11} enters basis in the second row. Product form of the inverse yields

$$B_1^{-1} = E_0 B_0^{-1} = \begin{bmatrix}1 & -\frac{4}{3} & 0\\ 0 & 2 & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\frac{1}{3} & 0 & 0\\ 0 & \frac{1}{8} & 0\\ 0 & 0 & \frac{1}{6}\end{bmatrix} = \begin{bmatrix}\frac{1}{3} & -\frac{1}{6} & 0\\ 0 & \frac{1}{4} & 0\\ 0 & 0 & \frac{1}{6}\end{bmatrix}.$$

Now

$$c_B B_1^{-1} = \begin{bmatrix}1 & 1 & 1\end{bmatrix}\begin{bmatrix}\frac{1}{3} & -\frac{1}{6} & 0\\ 0 & \frac{1}{4} & 0\\ 0 & 0 & \frac{1}{6}\end{bmatrix} = \begin{bmatrix}\frac{1}{3} & \frac{1}{12} & \frac{1}{6}\end{bmatrix}.$$

Column generation is used again to determine which pattern should enter the basis. For the current dual value $c_B B_1^{-1}$, a pattern described by y_1, y_2 and y_3 has a value $z_j - c_j$ as follows:

$$c_B B_1^{-1}\begin{bmatrix}y_1\\y_2\\y_3\end{bmatrix} - 1 = \frac{1}{3}y_1 + \frac{1}{12}y_2 + \frac{1}{6}y_3 - 1.$$

Equation (6) shows the equivalent knapsack problem.

$$\begin{aligned}\max\ t &= \frac{1}{3}y_1 + \frac{1}{12}y_2 + \frac{1}{6}y_3 - 1,\\ \text{subject to } 7y_1 + 3y_2 + 4y_3 &\leq 12,\\ y_i &\geq 0 \text{ and integer.}\end{aligned} \tag{6}$$

The knapsack problem in Equation (6) is then solved using dynamic programming. To solve this problem, start from the initial stage ($i = 1$) to the final stage or moving forward.

Case 1: $7y_1 \leq 12$.

Using dynamic programming method, the optimal solution of the case $7y_1 \leq 12$ is $y_1 = 1$ with $t_1 = t = \frac{1}{3}$.

Case 2: $7y_1 + 3y_2 \leq 12$.

Using dynamic programming method yields the optimal solution $y_1 = 1$, $y_2 = 1$ for the case $7y_1 + 3y_2 \leq 12$ with $t_2 = t = \frac{5}{12}$.

Case 3: $7y_1 + 3y_2 + 4y_3 \leq 12$.

Using dynamic programming method, the optimal solution of the case $7y_1 + 3y_2 + 4y_3 \leq 12$ is $y_1 = 1$, $y_2 = 0$, $y_3 = 1$ with $t_3 = t = \frac{1}{2}$.

The next step involves solving the knapsack problem with a width constraint, utilizing the objective function values derived from the length-constrained knapsack problem. Equation (7) displays the resulting formulation of the knapsack problem as follows:

$$\begin{aligned}\max\ t &= \frac{1}{3}y_1 + \frac{5}{12}y_2 + \frac{1}{2}y_3 - 1,\\ \text{subject to } 3y_1 + 4y_2 + 5y_3 &\leq 10,\\ y_i &\geq 0 \text{ and integer.}\end{aligned} \tag{7}$$

To solve Equation (7), the procedure begins at the initial stage ($i = 1$) and continues forward until the final stage.

Case 1: $3y_1 \leq 10$.

Using dynamic programming method produces the optimal solution $y_1 = 3$ for the case $3y_1 \leq 10$ with $t = t_1 = 0$.

Case 2: $3y_1 + 4y_2 \leq 10$.

Using dynamic programming method, the optimal solution for the case $3y_1 + 4y_2 \leq 10$ is $y_1 = 2$, $y_2 = 1$ with $t = t_2 = \frac{1}{12}$.

Case 3: $3y_1 + 4y_2 + 5y_3 \leq 10$.

Using dynamic programming method yields the optimal solution $y_1 = 2$, $y_2 = 1$, $y_3 = 0$ for the constraint $3y_1 + 4y_2 + 5y_3 \leq 10$ with $t = t_3 = \frac{1}{12}$.

From Cases 1, 2, and 3 obtained under the width constraints, the maximum value of the objective function t occurs in Case 2, where $t = t_3 = \frac{1}{12}$, therefore the optimal solution to the knapsack problem in Equation (7) is given by integer values $y_1 = 2$, $y_2 = 1$ and $y_3 = 0$ which yield $t = \frac{1}{12}$. Furthermore, since the knapsack solution under the width constraints corresponds to the solution under the length constraints, the updated values are obtained as $y_1 = 3$, $y_2 = 1$ and $y_3 = 0$.

This corresponds to pattern 7 and variable x_7. Therefore, the value of $z_7 - c_7$ is 1/12. However, this cutting pattern is not optimal, because $t = 1/12$. Next, we will look for a pattern that will provide benefits by incorporating the obtained pattern into the basis. For this, a new right-hand side and column need to be created.

$$\text{The new } x_7 \text{ column} = B_1^{-1}\begin{bmatrix}3\\1\\0\end{bmatrix} = \begin{bmatrix}\frac{1}{3} & -\frac{1}{6} & 0\\ 0 & \frac{1}{4} & 0\\ 0 & 0 & \frac{1}{6}\end{bmatrix}\begin{bmatrix}3\\1\\0\end{bmatrix} = \begin{bmatrix}\frac{5}{6}\\ \frac{1}{4}\\ 0\end{bmatrix}.$$

$$\text{The new right hand side} = B_1^{-1}b = \begin{bmatrix} \frac{1}{3} & -\frac{1}{6} & 0 \\ 0 & \frac{1}{4} & 0 \\ 0 & 0 & \frac{1}{6} \end{bmatrix} \begin{bmatrix} 18 \\ 22 \\ 14 \end{bmatrix} = \begin{bmatrix} \frac{14}{3} \\ \frac{22}{4} \\ \frac{14}{6} \end{bmatrix}.$$

Ratio test shows that x_7 enters basis in the first row. The new basic variables are $\{x_7, x_{11}, x_{33}\}$. Using the product form of the inverse is obtained

$$B_2^{-1} = E_0 B_1^{-1} = \begin{bmatrix} \frac{6}{5} & 0 & 0 \\ -\frac{3}{10} & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{3} & -\frac{1}{6} & 0 \\ 0 & \frac{1}{4} & 0 \\ 0 & 0 & \frac{1}{6} \end{bmatrix} = \begin{bmatrix} \frac{2}{5} & -\frac{1}{5} & 0 \\ -\frac{1}{10} & \frac{3}{10} & 0 \\ 0 & 0 & \frac{1}{6} \end{bmatrix}.$$

Now

$$c_B B_2^{-1} = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \frac{2}{5} & -\frac{1}{5} & 0 \\ -\frac{1}{10} & \frac{3}{10} & 0 \\ 0 & 0 & \frac{1}{6} \end{bmatrix} = \begin{bmatrix} \frac{3}{10} & \frac{1}{10} & \frac{1}{6} \end{bmatrix}.$$

Once again, the column generation technique is applied to identify the pattern that will enter the new basis. Using the current dual values $c_B B_1^{-1}$, a pattern expressed by y_1, y_2 and y_3 is determined to have the value $z_j - c_j$ as

$$c_B B_1^{-1} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} - 1 = \frac{3}{10}y_1 + \frac{1}{10}y_2 + \frac{1}{6}y_3 - 1.$$

Equation (8) shows the equivalent knapsack problem.

$$\begin{aligned} \max\ t &= \frac{3}{10}y_1 + \frac{1}{10}y_2 + \frac{1}{6}y_3 - 1, \\ \text{subject to } & 7y_1 + 3y_2 + 4y_3 \le 12, \\ & y_i \ge 0 \text{ and integer.} \end{aligned} \tag{8}$$

The knapsack problem in Equation (8) is then solved using dynamic programming. The solution process begins at the initial stage ($i = 1$) and proceeds forward until the final stage.

Case 1: $7y_1 \le 12$.

Using dynamic programming method yields the optimal solution $y_1 = 1$ for the case $7y_1 \le 12$ with $t_1 = t = \frac{3}{10}$.

Case 2: $7y_1 + 3y_2 \le 12$.

Using dynamic programming method, the optimal solution of the case $7y_1 + 3y_2 \le 12$ is $y_1 = 0$, $y_2 = 4$ with $t_2 = t = \frac{2}{5}$.

Case 3: $7y_1 + 3y_2 + 4y_3 \le 12$.

Using dynamic programming method produces the optimal solution $y_1 = 0$, $y_2 = 0$, $y_3 = 3$ for the case $7y_1 + 3y_2 + 4y_3 \le 12$ with $t_3 = t = \frac{1}{2}$.

The next step is to solve the knapsack problem using the width constraint while incorporating the objective function values from each case obtained when solving the knapsack problem with the length constraint. Thus, the resulting form of the knapsack problem is shown in Equation (9) as follows:

$$\begin{aligned} \max\ t &= \frac{3}{10}y_1 + \frac{2}{5}y_2 + \frac{1}{2}y_3 - 1, \\ \text{subject to } & 3y_1 + 4y_2 + 5y_3 \le 10, \\ & y_i \ge 0 \text{ and integer.} \end{aligned} \tag{9}$$

To solve Equation (9), the procedure begins at the initial stage ($i = 1$) and continues forward until the final stage.

Case 1: $3y_1 \le 10$.

Using dynamic programming method produces the optimal solution $y_1 = 3$ for the case $3y_1 \le 10$ with $t = t_1 = -\frac{1}{10}$.

Case 2: $3y_1 + 4y_2 \le 10$.

Using dynamic programming method, the optimal solution of the case $3y_1 + 4y_2 \le 10$ is $y_1 = 2$, $y_2 = 1$ with $t = t_2 = 0$.

Case 3: $3y_1 + 4y_2 + 5y_3 \le 10$.

Using dynamic programming method, the optimal solution of the constraint $3y_1 + 4y_2 + 5y_3 \le 10$ is $y_1 = 2$, $y_2 = 1$, $y_3 = 0$ with $t = t_3 = 0$.

From cases 1, 2, and 3 obtained using width constraints, the maximum value of the objective function t occurs in case 2 with $t = 0$, so that the optimal solution of the knapsack problem from Equation (9) is obtained by integer solutions, namely $y_1 = 2$, $y_2 = 1$ and $y_3 = 0$ which results in $t = 0$. Then, because the knapsack solution with width constraints corresponds to the knapsack solution with length constraints, so that new values of y_1, y_2 and y_3 are $y_1 = 2$, $y_2 = 4$ and $y_3 = 0$.

Since the solution yields $t = 0$, this indicates that there are no additional promising patterns to enter the basis, meaning the current basic variables are already optimal. The resulting basic variables are $\{x_7, x_{11}, x_{33}\}$. To identify the basic variables in the optimal solution, the value of the right-hand side is calculated as follows:

$$B_2^{-1}b = \begin{bmatrix} \frac{2}{5} & -\frac{1}{5} & 0 \\ -\frac{1}{10} & \frac{3}{10} & 0 \\ 0 & 0 & \frac{1}{6} \end{bmatrix} \begin{bmatrix} 18 \\ 22 \\ 14 \end{bmatrix} = \begin{bmatrix} \frac{14}{5} \\ \frac{24}{5} \\ \frac{7}{3} \end{bmatrix}.$$

Thus, the optimal solution for the cutting-stock problem described above is as follows: $x_7 = 14/5$, $x_{11} = 24/5$ and $x_{33} = 7/3$. The integer solution is obtained by rounding up, namely $x_7 = 3$, $x_{11} = 5$ and $x_{33} = 3$. The demand for 18 sheets of 7 ft×3 ft size, 22 sheets of 3 ft × 4 ft and 14 sheets of 4 ft × 5 ft size can be met by cutting standard 12 ft × 10 ft steel plate. The cutting process is performed by adjusting the cutting blade. The steel plate is first cut along the length and then along the width. The required pieces cannot be rotated during the cutting process, resulting in the following configuration:

1. Cut three sheets of standard steel plate measuring 12 ft × 10 ft. First, cut two 10 ft sheets with a width of 3 ft and one 4 ft sheet, then one 12 ft sheet with a length of 7 ft and one 3 ft sheet, resulting in 3 sheets of 7 ft × 3 ft and 1 sheet of 3 ft × 4 ft for each standard 12 ft × 10 ft size. The total produced is 9 sheets of 7 ft × 3 ft and 3 sheets of 3 ft × 4 ft.

2. Cut 5 sheets of standard 12 ft × 10 ft steel plate. First, cut 3 sheets of 10 ft with a width of 3 ft and 1 sheet of 4 ft, then 2 sheets of 12 ft with a length of 7 ft and 4 sheets of 3 ft, so that one gets 2 sheets of 7 ft × 3 ft and 4 sheets of 3 ft × 4 ft for each standard 12 ft × 10 ft size. The total produced is 10 sheets of 7 ft × 3 ft and 20 sheets of 3 ft × 4 ft.
3. Cut 3 sheets of standard 12 ft × 10 ft steel plate. First, cut 2 sheets of 10 ft × 5 ft steel plate, then 3 sheets of 12 ft × 4 ft steel plate, resulting in 6 sheets of 4 ft × 5 ft steel plate for each standard 12 ft × 10 ft steel plate. This yields a total of 18 sheets of 4 ft × 5 ft steel plate.

5. CONCLUSIONS

This paper presents an approximation method for solving the rectangular cutting-stock problem, a class of NP-hard optimization problems that arises frequently in manufacturing, logistics, and material processing industries. The proposed approach is suitable for large-scale and practical instances where exact methods become infeasible. In addition to its performance benefits, the proposed method is relatively simple to implement and flexible enough to accommodate common industrial constraints, such as fixed sheet sizes and heterogeneous item dimensions. This makes it a practical alternative for real-world applications where rapid decision-making is essential. Despite these strengths, the method remains an approximation and does not guarantee optimality in all cases. Future research may focus on integrating the proposed approach with metaheuristic or hybrid optimization techniques to further improve solution quality. Extensions to handle additional constraints – such as item rotations or stochastic demand – also represent promising directions for further study

6. ACKNOWLEDGEMENT

The authors would like to thank the anonymous reviewers for their valuable feedback in enhancing the presentation of this paper.

REFERENCES

[1] John E Beasley. An exact two-dimensional guillotine cutting algorithm. *Operations Research*, 33(4):847–859, 1985.

[2] Hao Zhang, Shaowen Yao, Qiang Liu, Lijun Wei, Libin Lin, and Jiewu Leng. An exact approach for the constrained two-dimensional guillotine cutting problem with defects. *International Journal of Production Research*, 61(9):2986–3003, 2023.

[3] Dianjian Wu and Chunping Yan. Balanced approach for the two-dimensional rectangular guillotine cutting stock problem with setup cost. In *IOP Conference Series: Materials Science and Engineering*, volume 452, page 022087. IOP Publishing, 2018.

[4] François Clautiaux, Antoine Jouglet, Jacques Carlier, and Aziz Moukrim. A new constraint programming approach for the orthogonal packing problem. *Computers & Operations Research*, 35(3):944–959, 2008.

[5] Didier Fayard, Mhand Hifi, and Vassilis Zissimopoulos. An efficient approach for large-scale two-dimensional guillotine cutting stock problems. *Journal of the Operational Research Society*, 49(12):1270–1277, 1998.

[6] André Soares Velasco and Eduardo Uchoa. Improved state space relaxation for constrained two-dimensional guillotine cutting problems. *European Journal of Operational Research*, 272(1):106–120, 2019.

[7] Mir-Bahador Aryanezhad, Nima Fakhim Hashemi, Ahmad Makui, and Hasan Javanshir. A simple approach to the two-dimensional guillotine cutting stock problem. *Journal of Industrial Engineering International*, 8(1):21, 2012.

[8] Dianjian Wu and Guangyou Yang. A heuristic approach for two-dimensional rectangular cutting stock problem considering balance for material utilization and cutting complexity. *Advances in Materials Science and Engineering*, 2021(1):3732720, 2021.

[9] Sergey Polyakovskiy and Peter J Stuckey. A constraint programming solution to the guillotine rectangular cutting problem. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 33, pages 352–360, 2023.

[10] Paul C Gilmore and Ralph E Gomory. A linear programming approach to the cutting-stock problem. *Operations Research*, 9(6):849–859, 1961.

[11] Paul C Gilmore and Ralph E Gomory. A linear programming approach to the cutting stock problem—part II. *Operations Research*, 11(6):863–888, 1963.

[12] Weiping Pan. An exact algorithm for two-dimensional cutting problems based on multi-level pattern. *Graphical Models*, 133:101220, 2024.

[13] Glauber Ferreira Cintra, Flávio Keidi Miyazawa, Yoshiko Wakabayashi, and Eduardo C Xavier. Algorithms for two-dimensional cutting stock and strip packing problems using dynamic programming and column generation. *European Journal of Operational Research*, 191(1):61–85, 2008.

[14] Fabio Furini, Enrico Malaguti, Rosa Medina Durán, Alfredo Persiani, and Paolo Toth. A column generation heuristic for the two-dimensional two-staged guillotine cutting stock problem with multiple stock size. *European Journal of Operational Research*, 218(1):251–260, 2012.

[15] K Novianingsih, R Hadianti, and S Uttunggadewa. Column generation technique for solving two-dimensional cutting stock problems: method of stripe approach. *Journal of the Indonesian Mathematical Society*, pages 161–172, 2007.

[16] Sirirat Juttijudata and Phissanu Sudjarittham. Two-dimensional cutting stock problems with a modified column generation method. *Current Applied Science and Technology*, pages 217–225, 2020.

[17] Jianyu Long, Zhong Zheng, Xiaoqiang Gao, Panos M Pardalos, and Wanzhe Hu. An effective heuristic based on column generation for the two-dimensional three-stage steel plate cutting problem. *Annals of Operations Research*, 289(2):291–311, 2020.

[18] Massimo Bertolini, D Mezzogori, F Zammori, et al. Hybrid

heuristic for the one-dimensional cutting stock problem with usable leftovers and additional operating constraints. *International Journal of Industrial Engineering Computations*, 15(1):149–170, 2024.

[19] Qiulian Chen and Yan Chen. Heuristics for the two-dimensional cutting stock problem with usable leftover. *Intelligent Data Analysis*, 28(2):591–611, 2024.

[20] Wayne L Winston. *Operations research: applications and algorithm*. Thomson Learning, Inc., 2004.

[21] Hamdy A Taha. *Operations research: an introduction*. Pearson Education India, 2013.