國　立　中　正　大　學

企業管理研究所碩士論文

Dynamic Programming Algorithm for Cutting Stock Problem with Multiple-Size Rectangles

研 究 生： 何建進

指導教授： 葉丁鴻 博士

中華民國 107 年 6 月

致謝

在中正兩年的歲月裡，最幸運的事情莫過於能在葉老師門下學習了。

在學生撰寫論文的路上，葉老師總是輕鬆活潑將複雜困難的數學問題教的簡單、耐心熱忱地不斷重複複雜觀念的講解，對於學生的想法亦是給予十足的鼓勵與包容，讓學生能找到屬於自己的天空發揮所學，完成碩士論文。論文之外的指導更是影響學生甚鉅，老師樂觀開朗的處世之道與淡泊名利的人生哲學無一不令學生敬佩而欲效法；而職涯方向與留學選擇的建議與幫忙，就像茫茫大海上的一盞明燈般給了學生一個光明的指引。學生心中只有滿滿的感激。

感謝口試委員阮金聲老師、陳明德老師的建議與提點，使論文更臻完美；感謝父母在我選擇碩班時的支持；感謝中正同學的陪伴與幫助；感謝一路陪我度過許多不同人生階段的遠方友人的陪伴與加油。因為有你們，使我更加強大。

何建進　謹誌於
中正大學管理學院
中華民國一零八年一月

摘要

原物料切割問題相關研究在許多產業中被廣泛的應用著，諸如造紙、鋼鐵、玻璃、膠捲等產業，皆與之息息相關。其不外乎提供切割作業過程中一良好的切割排樣使邊料浪費最少或使需求產出最大。此研究基於單尺寸矩形最簡化最佳排樣的動態規劃演算法發展出多尺寸矩形最簡化最佳排樣動態規劃演算法，並提出一完整的四階段演算法使多種需求尺寸矩形的生產者可以快速獲得最簡化最佳生產排樣進行生產。演算法包括：(1)規範尺寸運算，目的以獲得一集合包括所有可能規範尺寸來有效提升動態規劃演算法遞迴效率；(2)動態規劃演算法執行，目的以獲得最簡化最佳排樣；(3)記錄追蹤演算法執行，目的以獲得執行動態規劃演算法後的資料記錄來進行排樣；(4)規範排樣調整，目的以獲得規範排樣使排樣更趨簡單明瞭易於實務操作。簡單實例說明此演算法能有效解決多尺寸矩形原物料切割問題並獲得最簡化最佳生產排樣。

關鍵字：　二維原物料切割問題、動態規劃、多尺寸矩形

Abstract

To minimize the waste of trim loss, studies of cutting stock problem (CSP) have been discussed in very diverse industries such as paper, steel, grass, film etc. In this study, we adapt a dynamic programming algorithm of constrained equal rectangle CSP for multiple-size rectangle CSP and propose an algorithm which can guarantee the simplest optimal pattern while cutting the multiple-size stock pieces for customers. The proposed algorithm includes 4 stages: (1) Normal length generating, which generates a set with more efficient step off points for dynamic programming calculation. (2) Dynamic programming algorithm running, which results in the simplest optimal pattern of multiple-size rectangle CSP. (3) Section back tracking, which traces the resulting records back to arrange the layout. (4). Normal pattern justifying, which adjusts the simplest optimal pattern to a normal pattern. The illustrative example shows our proposed algorithm is capable to solve multiple-size CSP.

Keywords: 2-Dimensional cutting stock problem, Dynamic programming, Multiple-size rectangles

Contents

List of Figures

List of Tables

Chapter 1 Introduction

1.1. Motivation

While manufacturers cut stock sheets into smaller pieces for customer's demand, some waste occur. Waste control is crucial for company because of its surprising cost of trim loss when the number of order comes to a large quantities. To minimize the waste from cutting process, there are lots of methods proposed in the past six decades so far. This type of problems for finding optimal cutting patterns also called cutting stock problem (CSP). Cutting stock problem have been discussed in wide variety of manufacturing industries, such as paper, steel, grass, metal, film, LCD industry etc.

The motivation of this study originates from a paper by Cui (2005) who proposed a dynamic programming algorithm(here after DPA) for finding the simplest optimal guillotine cutting pattern of equal rectangles. Although stage cutting (will be illustrated in chapter 1.2.3) has been discussing for many decades, papers discussed cutting stage are all stage-cut priority, which means it may need to compromise its utilization of cutting pattern and will have a lot of wastes. In Cui's paper, he used dynamic programming to propose a utilization priority algorithm for finding the simplest optimal pattern of single rectangle. This algorithm has a faster calculating time than branch-and-bound algorithm(Agrawal, 1993; Cui & Zhou, 2002) in cutting stock problem of equal rectangles (CSPER). However, the stock pieces size may be two or above in reality. In our study, we developed an algorithm modified from that DPA to give a solution fulfilling multiple stock piece sizes CSP.

1.2. Literature Review

Cutting stock problem (CSP) is one of the famous NP-hard problems in Operations Research. Since the first use of integrate linear programming for solving CSP proposed by Gilmore and Gomory in 1961, there are enormous discussions came up during these years. We will brief two kinds of solutions in two-dimensional cutting stock problem at first: Exact solution and Heuristic. The following section we focus on our study scope: 2-D stage cut limitation of cutting pattern.

1.2.1. Exact Solution

Using exact solution to solve cutting stock problem can obtain the optimal solution. There're several critical methods showed as below:

Mathematical programming

It's common to see that using mathematical programming to solve cutting stock problem. For mathematical programming, in general, objective function and constraints need to be constructed before calculating. Roughly speaking, the principle of this method is to find out the optimal solution from searching corner solutions within constraints. This form was first applied by P. C. Gilmore and Gomory (1961) to deal with knapsack problem. Few years later, they extended knapsack function to cope with multi-staged two-dimensional cutting stock problem(P. Gilmore & Gomory, 1965). These works done by Gilmore and Gomory became a paradigm in CSP and plenty of scholars developed solving models based on their jobs since then, see for example Dyckhoff (1981), Farley (1992), De Carvalho (2002).

When finding optimal solution of CSP, due to the various shapes of order size (stock piece) and row material (stock rectangle), the number of possible cutting patterns would be extremely large. To avoid enumerating every possible cutting patterns literally, P. C. Gilmore and Gomory (1961) figured out a method named Column Generation (also called Decomposition Principle) to give an efficient way to overcome computationally difficulty. The logic behind this is to separate constraints into two types. First type constraints are defined as master problem which is not easy to solve directly. Second type constraints are defined as sub problem which possess special properties and easy to find the solutions. Based on duality theorem, we can easily solve sub problem at first, substitute these solutions into master problem to improve the objective value until there's no more improvement of objective value. For papers developing the following column generation linear programming model to solve CSP, see for example Haessler (1980), Zak (2002).

Dynamic programming

The initial concept of dynamic programming can be traced back to the beginning of the 1950s. Bellman (1952) proposed Principle of Optimality while studying multistep decision process. The method is to decompose multivariable problem into several stages, each stage comes to a single-variable sub problem. Every stage is linked by recursive relation. It's easier to compute when solving only one variable at one moment. After solving a series of sub problems, the solution of multivariable problem is obtained ultimately.

In cutting stock problem, this method was first adopted by P. Gilmore and Gomory (1966). They discussed one-dimensional and two-dimensional CSP respectively. This fundamental dynamic programming model of CSP provided a basic idea to the following studies. Herz (1972) suggested Gilmore and Gomory's models can only consider canonical dissections cut which will have more efficiency while computing. Based on Herz's work, Beasley (1985a) proposed the concept of normal lengths which means only searching the possible size in the stock rectangle. This algorithm prevents irrelevant computations and increase the efficiency of computing process significantly. Subsequently, Cintra et al. (2008) mixed column generation and dynamic programming to decrease the number of normal lengths and the complexity of computation.

1.2.2. Heuristic

Approaches which are implemental for finding immediate solution can be called heuristic. Though the solution may not be the optimal solution, it provides a way to solve highly complicated problem simply in practice or offers a fast computation process to obtain a feasible solution. Owing to the advance of today's computational technology, the possibility to develop heuristics for practical use has increased largely.

Regarding to cutting stock problem, Christofides and Whitlock (1977) first presented the tree-search algorithm to solve two-dimensional CSP. In their paper, they used dynamic programming for finding unconstrained problem's solution and applied a node evaluation method to produce upper bounds during the search, both of which limit the size of the tree search. Next, Beasley (1985b) adopted tree-search procedure to solve two-dimensional non-guillotine CSP. He developed a Lagrangean relaxation of a zero-one integer programming formulation to provide a bound in the tree search.

Morabito et al.(1992) also used tree-search to solve related problem. However, they combined two classical strategies which are depth-first and hill-climbing to search the nodes in the tree search (i.e. DH algorithm). Another approach called KD algorithm, proposed by Fayard and Zissimopoulos (1995), solves a sequence of one-dimensional knapsack problems to obtain optimal strips and uses these optimal generated strips to find out the optimal subset on two-dimensional plate. Later, Hifi (1997) combined DH and KD algorithm (i.e. DH/KD algorithm) to offer a more efficient way to solve CSP. For other approaches such as particle swarm optimization, ant colony optimization, tabu search algorithm, simulated annealing algorithm, genetic algorithm etc., see for example Shen et al.(2007), Levine and Ducatelle (2004), Alvarez-Valdés and Parajón (2002), Chen et al.(1996) and Leung et al.(2001).

1.2.3. 2-D Stage Cut Limitation of Cutting Pattern

In practice, it's important to obtain a cutting pattern which has the least waste and is easy to cut off meanwhile. To satisfy this goal, P. Gilmore and Gomory (1965) first discussed the issues of cutting complexity in their paper. They proposed a linear programming model of two-stage guillotine cut to take the cutting complexity into account. For a guillotine cutting pattern, we can obtain the stock pieces by cutting off the stock rectangle from one edge across to the opposite edge, and the cutting stage means the needed times of cut to achieve all stock pieces, as shown in Figure 1.1.

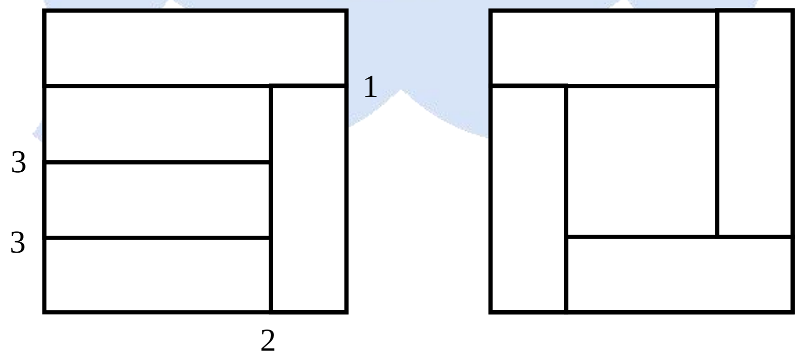


Figure 1.1. (a) Three-stage guillotine pattern and (b) non-guillotine pattern (also called nested pattern).

Since then, many scholars developed their models considering cutting stage. See for example Hifi and Roucairol (2001), Belov et al.(2006) and Furini et al.(2013). Cui (2005) used dynamic programming to propose a utilization priority algorithm for finding the simplest optimal pattern of single rectangle. In his paper, the algorithm can guarantee the minimum number of sections cutting pattern in CSPER.

Chapter 2 Problem Definition

2.1. Problem Description

In this thesis, we present a dynamic programming algorithm to solve cutting stock problems, which can generate the simplest optimal pattern (will be illustrated in chapter 2.2) by considering multiple stock piece sizes. According to the typology of Dyckhoff (1990), this type of CSP corresponds to 2/B/O/R or is classified as 2-D Single Large Object Placement Problem by Wäscher et al.(2007).

For our study, there are several assumptions must be followed:

1. For stock rectangle $A = (L, W)$ and stock piece i. The lengths L, l_i and the widths W, w_i are integers for $i = 1, 2, \ldots, n$. The sizes of stock rectangle and stock piece must be given.
2. Only guillotine cut is allowed to cut the stock pieces off stock rectangle, which means the stock pieces can only be obtained by cutting off the stock rectangle from one edge across to the opposite edge.
3. The orientation of stock pieces is fixed, which means a stock piece of length x and width $y (x \neq y)$ is not equal to a stock piece of length y and width x.
4. Cut are made to be infinitely thin, which means the waste of cut line can be ignored.
5. The exact demands of stock pieces are not considered.
6. The proportion of different stock piece sizes within a stock rectangle is not considered in the output pattern, which means it's possible that the simplest optimal pattern is only with one stock piece size.

2.2. Notations and Definitions

The notations used in this study are summarized in Table 2.1. Most of them will be re-introduced when we use it for the first time.

Table 2.1. Notations.

Notation	Type	Meanings
l_i	Parameter	Length of i th stock piece size, $i = 1, 2, .., S$.
w_i	Parameter	Width of i th stock piece size, $i = 1, 2, .., S$. $l_i \geq w_i$.
S	Parameter	All we have S sizes of stock piece.
L	Parameter	Length of stock rectangle.
W	Parameter	Width of stock rectangle. $L \geq W$.
A	Parameter	The set of normal lengths. $A = \{a_1, \dots, a_M\}$, a_i is the combination number of stock pieces length and widths, where $a_1 = 0$ and $a_{i+1} > a_i$ for $1 \leq i \leq M$.
a_i	Parameter	Normal length. The linear combination of $\sum_{i=1}^{S} m_i l_i + k_i w_i$.
$p(x)$	Parameter	Return the maximum normal length no longer than x.
$int(x)$	Parameter	Return the maximum integer no larger than x.
j	Categorized variable	The basis of strip direction. $j \in J$, $J = \{w, l\}$.
k	Categorized variable	The direction of strip. $k \in K$, $K = \{X, Y\}$.
$F(x, y)$	Decision variable	The maximum value yielded by rectangle $x \times y$.
$Q(x, y)$	Decision variable	Policy. Records the best value cut way on rectangle $x \times y$.
$G(x, y)$	Decision variable	Records the minimum number of sections yielded by rectangle $x \times y$.

The related definitions are briefly described as below:

Definition 1. Strips and sections

According to Cui (2005), A strip includes one or more stock pieces. In his paper, he only considered single size of stock piece. Here we revise some notations to adapt to our algorithm for multiple-size CSP. As shown in Figure 2.1, which is an example illustrates four possible cutting pattern with every stock piece size. Figure 2.1(a) shows a strip which its stock piece width parallel to X-direction and the strip type jk is denoted as wX, (b) a strip which its stock piece width parallel to Y-direction and the strip type jk is denoted as wY, (c) a strip which its stock piece length parallel to X-direction and the strip type jk is denoted as lX, (d) a strip which its stock piece length parallel to Y-direction and the strip type jk is denoted as lY.

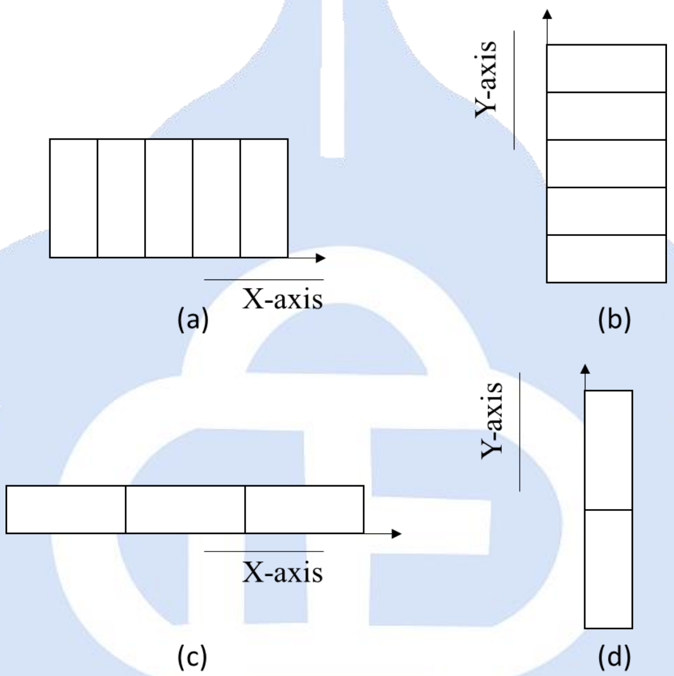
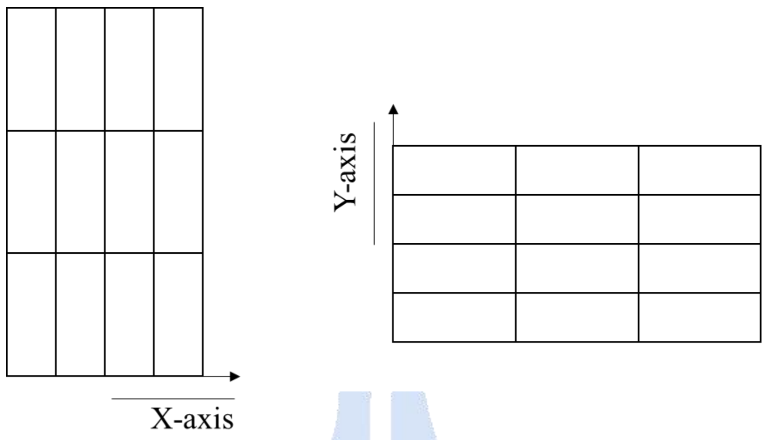


Figure 2.1. (a)A strip which its type $jk = wX$, (b) a strip which its type $jk = wY$, (c) a strip which its type $jk = lX$ and (d) a strip which its type $jk = lY$.

A section consists of one or several strips of the same length. There is no difference between a section which its stock piece width parallel to X-direction and a section which its stock piece length parallel to Y-direction. Thus, there are only two types of section directions. As shown in Figure 2.2. Figure 2.2(a) shows a section which its stock piece width parallel to X-direction and (b) a section which its stock piece width parallel to Y-direction.



(a) (b)

Figure 2.2. (a) A section with its stock piece width parallel to X-direction and (b) a section with its stock piece width parallel to Y-direction.

Definition 2. Normal lengths

Assume that the size of i th stock piece is $l_i \times w_i$, for $i = 1, 2, .., S$. x is a normal length if $x = \sum_{i=1}^{S} m_i l_i + k_i w_i$, m_i and k_i are all non-negative integers, and $x > 0$. That is, normal lengths are the linear combinations of every stock piece length and width. Here we note the vector of normal lengths as $A = \{a_1, \dots, a_M\}$, where $a_1 = 0$ and $a_{i+1} > a_i$ for $1 \le i \le M$.

Definition 3. Optimal patterns and the simplest optimal patterns

The optimal patterns guarantee the maximum utilization of a stock rectangle, which means the waste area will be the least among all possible cutting patterns. If an optimal pattern is of the minimum number of sections among all optimal patterns, it is defined as the simplest optimal pattern. As shown in Figure 2.3, (a) is one instance of all optimal patterns which its number of sections is 5 and (b) is the simplest optimal pattern among all optimal patterns which its number of sections is 4.

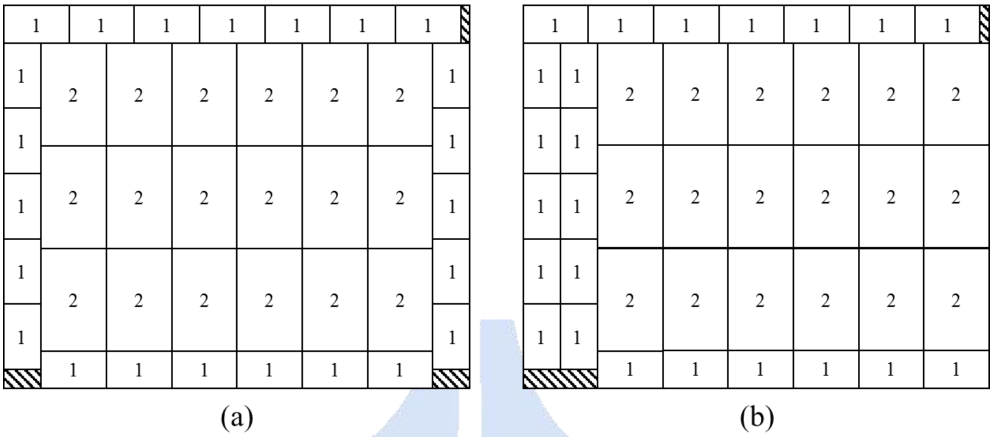


Figure 2.3. (a) An optimal pattern with 5 sections and (b) a simplest optimal pattern with 4 sections.

Definition 4. Normal patterns of multiple stock piece

In Cui (2005), he defined a normal pattern as a pattern which its sections are top-right justified if the stock piece width parallel to Y-direction and sections are bottom-left justified if the stock piece width parallel to X-direction. All odd sections are in the same direction and all even sections are in the opposite direction. Section 1 is possible to be the direction which its stock piece width parallel to Y-direction or X-direction. As shown in Figure 2.4. Figure 2.4 is a normal pattern that its section 1's stock piece width parallel to Y-direction.

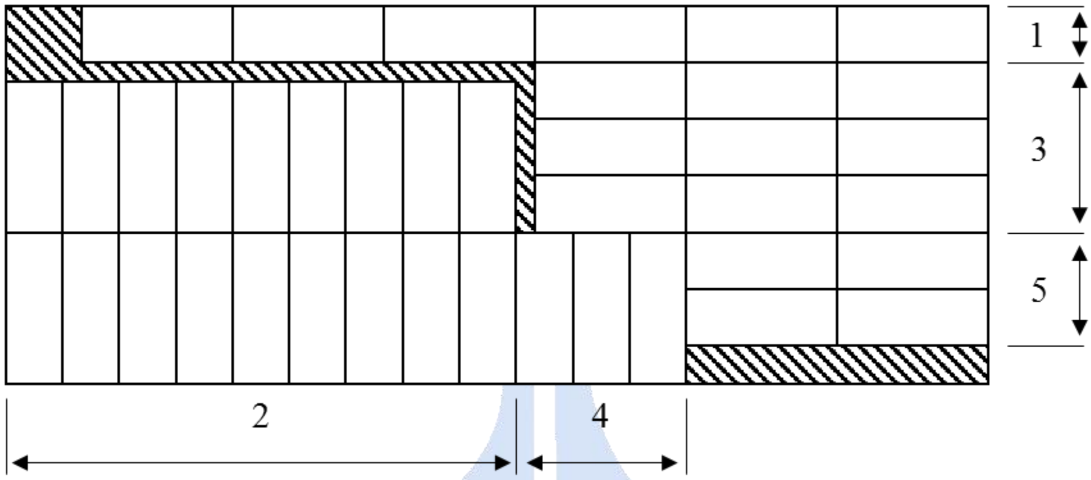
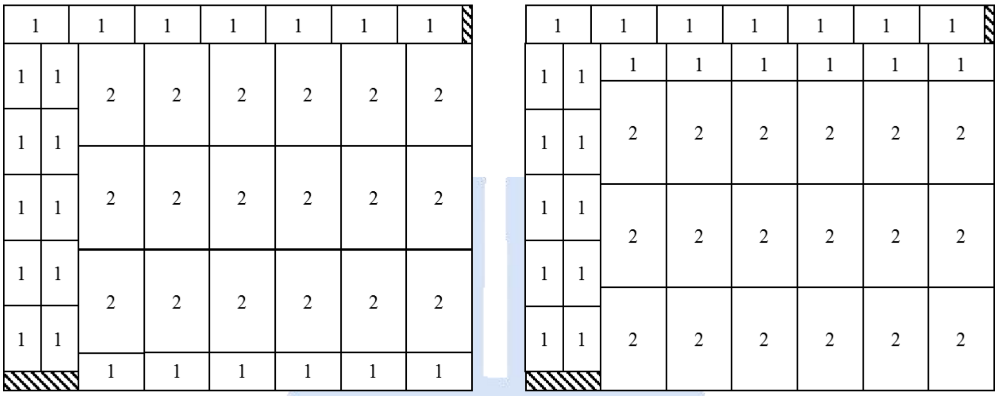


Figure 2.4. Normal pattern of single stock piece cutting stock problem (Cui, 2005).

It has been proofed that any multi-section pattern can be transformed to a normal pattern without decreasing the number of stock pieces or increasing the number of sections by Agrawal (1993) and Cui et al.(2002). When x_0 be the maximum normal length no longer than x and y_0 be the maximum normal length no larger than y, then $F(x_0, y_0) = F(x, y)$, which means stock rectangle $x \times y$ can be trimmed to rectangle $x_0 \times y_0$ without decreasing the number of stock pieces.

In our study, the pattern of multiple-size CSP will be more complicated than CSPER. Thus, we ignore the direction of the section and follow a simple rule: Gather equal size stock piece. According to previous study, we can shift the pattern without decreasing the number of stock pieces and without increasing the number of sections. After we generated the simplest optimal pattern, we can justify the simplest optimal pattern into a normal pattern instinctively which its equal size stock piece will gather together. As shown in Figure 2.5, (a) is a simplest optimal pattern of multiple-size CSP and (b) is a normal pattern justified from (a).



(a) (b)

Figure 2.5. (a) A simplest optimal pattern with 4 sections and (b) a normal pattern with 4 sections.

Chapter 3 Research Methodology

3.1. Model Formulation

Before describing our proposed formulation, let's first review Cui (2005)'s recursion of dynamic programming in CSPER:

$$F(x,y) = max\{F(x,y-l) + int(x/w), F(x-l,y) + int(y/w)\} \quad (1)$$

Formula (1) represents that two possible cuts may determine the pattern on stock rectangle $x \times y$:

Cut 1. As shown in Figure 3.1(a), lay an strip with strip type $jk = wX$ along the upper side of rectangle $x \times y - l$. The strip includes $int(x/w)$ stock pieces. The new rectangle $x \times y$ contains $F(x,y-l) + int(x/w)$ stock pieces.

Cut 2. As shown in Figure 3.1(b), lay an strip with strip type $jk = wY$ along the right side of rectangle $x - l \times y$. The strip includes $int(y/w)$ stock pieces. The new rectangle $x \times y$ contains $F(x-l,y) + int(y/w)$ stock pieces.

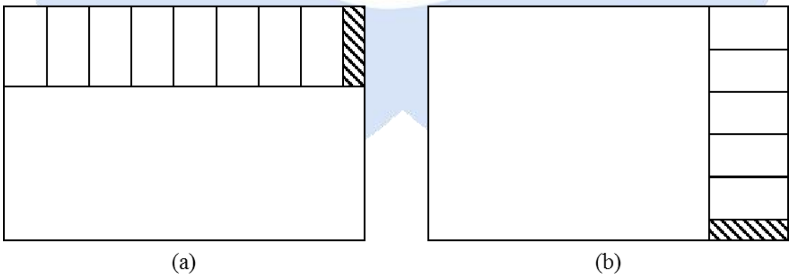


Figure 3.1. Two possible cuts on stock rectangle $x \times y$. (a) From $x \times y - l$. and (b) from $x - l \times y$.

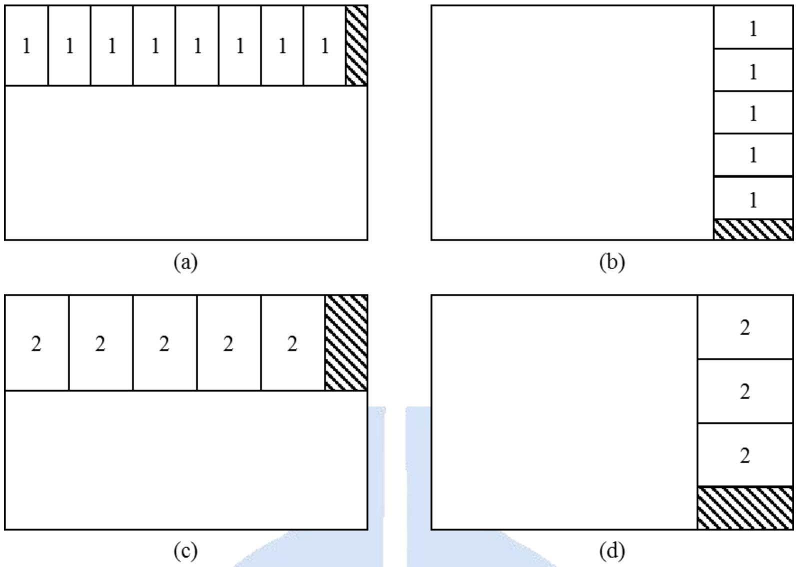


Figure 3.2. Two possible cuts with two sizes of stock piece on stock rectangle $x \times y$. (a) From $x \times (y - l_1)$, (b) from $(x - l_1) \times y$, (c) from $x \times (y - l_2)$ and (d) from $(x - l_2) \times y$.

We revised Formula (1) to add another size of stock piece i into the possible cut set, for $i = 1, 2, .., S$. See the following recursion:

$$F(x, y) = \max_{1 \le i \le S}\{F(x, y - l_i) + int(x/w_i) \times l_i w_i, F(x - l_i, y) + int(y/w_i) \times l_i w_i\} \quad (2)$$

Formula (2) shows that every size of stock piece will have two possible cuts to lead the pattern on stock rectangle $x \times y$, which also means that the total number of possible cuts will be $2 \times S$. An example of two stock piece sizes is demonstrated in Figure 3.2. Figure 3.2(a) is Cut 1. of first stock piece size, (b) is Cut 2. of first stock piece size, (c) is cut 1. of second stock piece size and (d) is Cut 2. of second stock piece size. Because the number of different size stock pieces ($int(x/w_i)$ and $int(y/w_i)$ in Formula (2)) possess different weight from each other, we let the number of stock pieces multiply its area $l_i \times w_i$ to overcome this problem. It also means that the recursive function is to search the maximum utilization of using area by stock pieces. Hence, the value of

$F(x, y)$ turns out to be the area covered by stock pieces.

However, Formula (2) may miss some possible cuts to lead the pattern, so the optimal pattern may not be obtained in some cases (See an example of Formula (2) in Appendix A). Thus, we consider four possible cuts of each stock piece size and propose Formula (3):

Cut 1. As shown in Figure 3.3(a), lay i th stock piece size strip with strip type $jk = wX$ along the upper side of rectangle $x \times (y - l_i)$. The strip area is $int(x/w_i) \times l_i w_i$. The stock pieces cover an area of $F(x, y - l_i) + int(x/w_i) \times l_i w_i$ on the new rectangle $x \times y$.

Cut 2. As shown in Figure 3.3(b), lay i th stock piece size strip with strip type $jk = wY$ along the right side of rectangle $(x - l_i) \times y$. The strip area is $int(y/w_i) \times l_i w_i$. The stock pieces cover an area of $F(x - l_i, y) + int(y/w_i) \times l_i w_i$ on the new rectangle $x \times y$.

Cut 3. As shown in Figure 3.3(c), lay i th stock piece size strip with strip type $jk = lX$ along the upper side of rectangle $x \times (y - w_i)$. The strip area is $int(x/l_i) \times l_i w_i$. The stock pieces cover an area of $F(x, y - w_i) + int(x/l_i) \times l_i w_i$ on the new rectangle $x \times y$.

Cut 4. As shown in Figure 3.3(d), lay i th stock piece size strip with strip type $jk = lY$ along the right side of rectangle $(x - w_i) \times y$. The strip area is $int(y/l_i) \times l_i w_i$. The stock pieces cover an area of $F(x - w_i, y) + int(y/l_i) \times l_i w_i$ on the new rectangle $x \times y$.

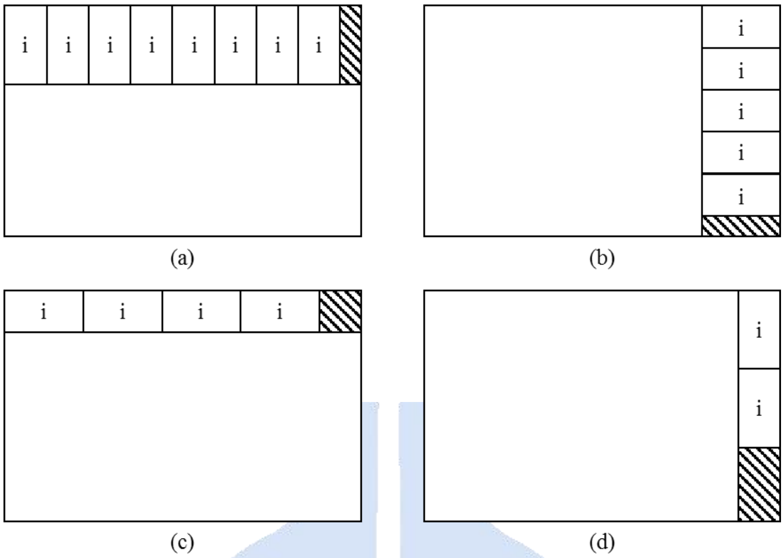


Figure 3.3. Four possible cuts with i th stock piece size strip on stock rectangle $x \times y$. (a) From $x \times (y - l_i)$, (b) from $(x - l_i) \times y$, (c) from $x \times (y - w_i)$ and (d) from $(x - w_i) \times y$.

To consider four possible cuts in every size of stock piece while calculating for the optimal pattern, the recursion is as below:

$$F(x,y) = \max_{1 \le i \le S} \begin{Bmatrix} F(x, y - l_i) + int(x/w_i) \times l_i w_i, F(x - l_i, y) + int(y/w_i) \times l_i w_i, \\ F(x, y - w_i) + int(x/l_i) \times l_i w_i, F(x - w_i, y) + int(y/l_i) \times l_i w_i \end{Bmatrix} \quad (3)$$

So far, Formula (3) guarantees the optimal pattern as we're looking for the solutions of muitiple-size CSP (See an example of Formula (3) in Appendix B). Regarding to obtaining the simplest optimal pattern, we must apply the following steps:

Let $n_{iwX} = F(x, y - l_i) + int(x/w_i) \times l_i w_i$, $n_{iwY} = F(x - l_i, y) + int(y/w_i) \times l_i w_i$, $n_{ilX} = F(x, y - w_i) + int(x/l_i) \times l_i w_i$ and $n_{ilY} = F(x - w_i, y) + int(y/l_i) \times l_i w_i$, for $i = 1, 2, \dots, S$. $G(x, y)$ is the number of sections of the optimal pattern in rectangle $x \times y$. $Q(x, y)$ records the path leads to $F(x, y)$. Let $F(x, y) = \max_{1 \le i \le S}\{n_{iwX}, n_{iwY}, n_{ilX}, n_{ilY}\}$.

$Q(x, y)$ and $G(x, y)$ are determined by bellows:

1. Let N be a set which its elements' value must equal to $F(x, y)$. See Formula (4). If the number of elements in set N which is $|N|$ is only one, let $Q(x, y) = (i, j, k)$ when the element in the set is n_{ijk}. For finding $G(x, y)$:

   Let $G(x, y) = G(x, y) + 1$ if $Q(x, y - l_i) \neq (i, j, k)$ when $n_{ijk} = F(x, y)$ and $j = w, k = X$. Let $G(x, y) = G(x, y) + 1$ if $Q(x, y - w_i) \neq (i, j, k)$ when $n_{ijk} = F(x, y)$ and $j = l, k = X$. Let $G(x, y) = G(x, y) + 1$ if $Q(x - l_i, y) \neq (i, j, k)$ when $n_{ijk} = F(x, y)$ and $j = w, k = Y$. Let $G(x, y) = G(x, y) + 1$ if $(x - w_i, y) \neq (i, j, k)$ when $n_{ijk} = F(x, y)$ and $j = l, k = Y$.

$$N = \{n_{ijk} | n_{ijk} = F(x, y), i \in I, j \in J, k \in K\} \tag{4}$$

2. If $|N|$ is greater than one, all elements in set N have equal utilization of using area in stock rectangle $x \times y$. We would like to choose the one with minimum number of sections. For finding the cut with minimum $G(x, y)$, we must add variable g_{ijk}:

   If $n_{ijk} = F(x, y)$ and $j = w, k = X$, let $g_{ijk} = G(x, y - l_i)$, let $g_{ijk} = g_{ijk} + 1$ if $Q(x, y - l_i) \neq (i, j, k)$. If $n_{ijk} = F(x, y)$ and $j = l, k = X$, let $g_{ijk} = G(x, y - w_i)$, let $g_{ijk} = g_{ijk} + 1$ if $Q(x, y - w_i) \neq (i, j, k)$. If $n_{ijk} = F(x, y)$ and $j = w, k = Y$, let $g_{ijk} = G(x - l_i, y)$, let $g_{ijk} =$

$g_{ijk} + 1$ if $Q(x - l_i, y) \neq (i, j, k)$. If $n_{ijk} = F(x, y)$ and $j = l, k = Y$, let $g_{ijk} = G(x - w_i, y)$, let $g_{ijk} = g_{ijk} + 1$ if $Q(x - w_i, y) \neq (i, j, k)$.

Now we let $G(x, y) = \min\{g_{ijk} | n_{ijk} = F(x, y)\}$ and give set G, let $G = \{g_{ijk} | g_{ijk} = G(x, y), i \in I, j \in J, k \in K\}$. If the number of elements in set G which is $|G|$ is only one, let $Q(x, y) = (i, j, k)$ when $g_{ijk} = G(x, y)$. If $|G|$ is greater than one, then $Q(x, y)$ can be any (i, j, k) when $g_{ijk} = G(x, y)$ because they all have the best utilization of using area and minimum number of sections. In this case, we will denote $Q(x, y)$ as the value follow first priority the least i, second priority $j = w > l$ and third priority $k = X > Y$ in our study.

3.2. Proposed Algorithm

To find the normal pattern of multiple-size CSP, the four stages of our proposed algorithm is shown in Figure 3.4:

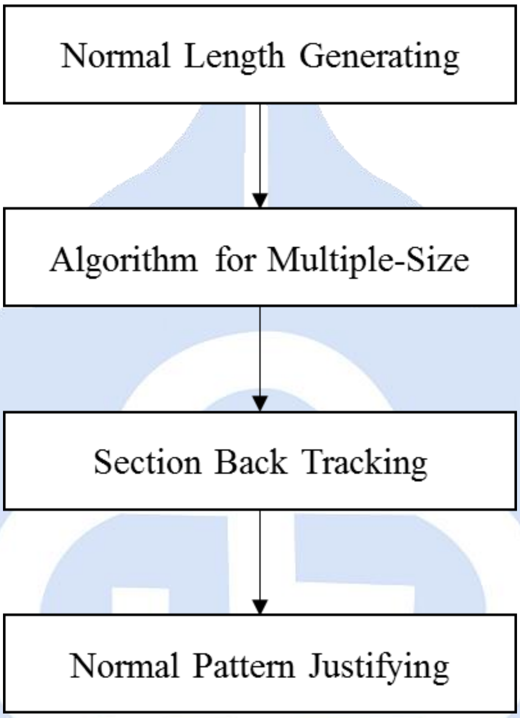


Figure 3.4. Flow chart of algorithm in four stages.

In the first stage, we generate the set of normal lengths $A = \{a_1, \ldots, a_M\}$ which is mentioned in Chapter 2.2 definition 2. In the second stage, we start searching the simplest optimal pattern along every length in set $A = \{a_1, \ldots, a_M\}$ in the algorithm revised from Algorithm A. proposed by Cui (2005) to fit for multiple-size CSP. The third stage is to trace the sections and strips records which have recorded during the second stage. The final stage will be executed by our intuition to output a normal pattern for easy-look purpose. We will explain the Algorithm for Multiple-Size in stage 2 and Section Back Tracking Algorithm in stage 3 in details.

1. Algorithm for Multiple-Size

Assume that the set of normal lengths $A = \{a_1, \ldots, a_M\}$ has been obtained from stage 1. $p(x)$ is the maximum normal length no greater than x. L_0 is the normalized stock rectangle length, that is, $L_0 = p(L) = a_M$. W_0 is the normalized stock rectangle width, that is, $W_0 = p(W) = a_N$. According to Chapter 2.2 Definition 4 and Chapter 3.1 Formula (3), the algorithm for generating simplest optimal pattern of multiple-size CSP is demonstrated as below:

Algorithm for Multiple-Size

Step 1. Let $F(x,y) = G(x,y) = Q(x,y) = 0$ for $x = a_1, \ldots,\ a_M, y = a_1, \ldots,$ a_N. $I = \{1, 2, .., S\}, J = \{w, l\}, K = \{X, Y\}$. Let $p = 0$.

Step 2. Let $p = p + 1$. Go to step 9 if $p > M$. Let $x = a_p$ and $x_{0iw} = p\{max(0, x - l_i)\}, x_{0il} = p\{max(0, x - w_i)\}, i \in I$. Let $q = 0$.

Step 3. Let $q = q + 1$. Go to step 2 if $q > N$. Let $y = a_q$ and $y_{0iw} = p\{max(0, y - l_i)\}, y_{0il} = p\{max(0, y - w_i)\}, i \in I$.

Step 4. Let $n_{jik} = 0, i \in I, j \in J, k \in K$.

If $y \geq l_i$, let $n_{iwX} = F(x, y_{0iw}) + int(x/w_i) \times l_i w_i$.

If $x \geq l_i$, let $n_{iwY} = F(x_{0iw}, y) + int(y/w_i) \times l_i w_i$.

If $y \geq w_i$, let $n_{ilX} = F(x, y_{0il}) + int(x/l_i) \times l_i w_i$.

If $x \geq w_i$, let $n_{ilY} = F(x_{0il}, y) + int(y/l_i) \times l_i w_i$.

Step 5. Let $F(x,y) = \max_{1 \leq i \leq S}\{n_{iwX}, n_{iwY}, n_{ilX}, n_{ilY}\}$.

Let $N = \{n_{ijk} | n_{ijk} = F(x,y), i \in I, j \in J, k \in K\}$.

If $|N| = 1$, go to step 6. If $|N| > 1$, go to step 7.

Step 6. Let $Q(x,y) = (i, j, k)$ where $n_{jik} = F(x,y), i \in I, j \in J, k \in K$.

Let $G(x,y) = max\{G(x, y_{0ij}), 1\}$ when $n_{jik} = F(x,y)$ and $k = X$.

Let $G(x,y) = max\{G(x_{0ij}, y), 1\}$ when $n_{jik} = F(x,y)$ and $k = Y$.

Let $G(x,y) = G(x,y) + 1$ if $Q\left(x, y_{0ij}\right) \neq (i,j,k)$ when $n_{jik} = F(x,y)$ and $k = X$. Let $G(x,y) = G(x,y) + 1$ if $Q\left(x_{0ij}, y\right) \neq (i,j,k)$ when $n_{jik} = F(x,y)$ and $k = Y$. Go to step 3.

Step 7. Let $g_{ijk} = G\left(x, y_{0ij}\right)$ when $n_{jik} = F(x,y)$ and $k = X$.

Let $g_{ijk} = G\left(x_{0ij}, y\right)$ when $n_{jik} = F(x,y)$ and $k = Y$.

Let $g_{ijk} = g_{ijk} + 1$ if $Q\left(x, y_{0ij}\right) \neq (i,j,k)$ when $n_{jik} = F(x,y)$ and $k = X$. Let $g_{ijk} = g_{ijk} + 1$ if $Q\left(x_{0ij}, y\right) \neq (i,j,k)$ when $n_{jik} = F(x,y)$ and $k = Y$.

Step 8. Let $G(x,y) = \min\left\{g_{ijk} \middle| n_{ijk} = F(x,y)\right\}$.

Let $G = \left\{g_{ijk} \middle| g_{ijk} = G(x,y), i \in I, j \in J, k \in K\right\}$.

If $|G| = 1$, let $Q(x,y) = (i,j,k)$ when $g_{ijk} = G(x,y)$.

If $|G| > 1$, let $Q(x,y)$ be any (i,j,k) when $g_{ijk} = G(x,y)$.

Let $g_{ijk} \in M, i \in I, j \in J, k \in K$. Go to step 3

Step 9. The maximum number of blanks is $F(L_0, W_0)$. Perform section-back-tracking to obtain the section layout of the simplest optimal pattern.

2. Section Back Tracking Algorithm

The purpose of this algorithm is to trace the direction and number of strips of each section.

Section Back Tracking Algorithm

Step 1. Let $x = L_0$ and $y = W_0$. Let m_k be the number of strips and q_k be the direction of the k th section. Let $k = 1$ and $m_1 = 0$. Let $q_k = (i, j, k)$ when $Q(x, y) = (i, j, k)$.

Step 2. Let $m_k = m_k + 1$ if $Q(x, y) = q_1$. Otherwise let $k = \mathrm{k} + 1$, $m_k = 1$ and $q_k = Q(x, y)$.

Step 3. Let $y = p(y - l_i)$ if $q_k = (i, w, X)$, let $y = p(y - w_i)$ if $q_k = (i, l, X)$; let $x = p(y - l_i)$ if $q_k = (i, w, Y)$, let $x = p(y - w_i)$ if $q_k = (i, l, Y)$.

Step 4. Go to step 5 if $F(x, y) = 0$, otherwise go to step 2.

Step 5. The number of strips and the direction of each section have been found. Output the simplest optimal pattern.

Chapter 4 Numerical Example

At the beginning of this chapter, we give a simple example to demonstrate the proposed algorithm. Secondly, we discuss the application of our study in Printed Circuit Board industry.

4.1. Illustrative Example for Proposed Algorithm

Finding the simplest optimal pattern when stock rectangle $A = (88{,}43)$, stock piece $i = 1, l_1 = 15, w_1 = 7; i = 2, l_2 = 6, w_2 = 6$ with area 105 and 36 respectively.

Stage 1. Generate normal lengths $A = \{a_1, \dots, a_M\}$. The normal lengths is shown as below:

Normal length:

$$A = \{a_1, \dots, a_M\} = \left\{\begin{matrix} 0, 6, 7, 12, 13, 14, 15, 18, 19, 20, 21, 22, 24, 25, 26, \\ 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, \\ 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, \\ 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, \\ 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, \\ 80, 81, 82, 83, 84, 85, 86, 87, 88 \end{matrix}\right\}$$

Normal width:

$$A = \{a_1, \dots, a_N\} = \left\{\begin{matrix} 0, 6, 7, 12, 13, 14, 15, 18, 19, 20, 21, 22, 24, 25, 26, \\ 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, \\ 40, 41, 42, 43 \end{matrix}\right\}$$

Stage 2. Run Algorithm for Multiple-Size. The maximum value of $F(88{,}43) = 3744$ with area utilization rate 98.9429% which is calculated by the following formula:

$$\text{Area utilization rate} = F(x, y)/(x \times y) \tag{5}$$

Stage 3. Run Section Back Tracking Algorithm. The policy is depicted in Table 4.1. and the simplest optimal pattern is shown in Figure 4.1.

Table 4.1. Section Back Tracking results(policy).

$F(x,y)$	Stock piece i	Direction jk	Number
$F(88,43) = 3744$	2	lY	7
$F(82,43) = 3492$	2	lY	7
$F(76,43) = 3240$	1	lX	5
$F(76,36) = 2715$	2	lY	6
$F(70,36) = 2499$	2	lY	6
$F(64,36) = 2283$	1	wY	5
$F(49,36) = 1758$	2	lX	8
$F(49,30) = 1470$	1	wX	7
$F(49,15) = 735$	1	wX	7

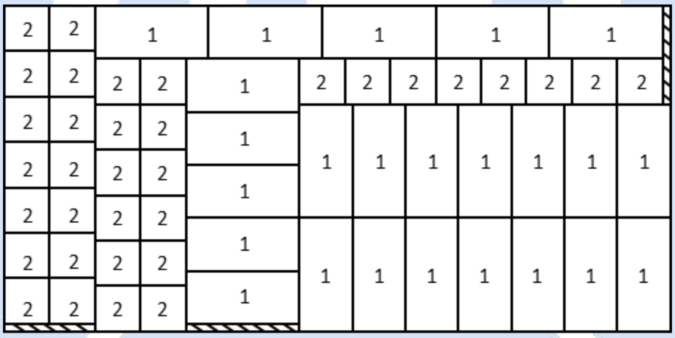


Figure 4.1. Simplest optimal pattern with 6 sections on stock rectangle $x \times y = 88 \times 43$.

Stage 4. Adjust simplest optimal pattern into normal pattern. The transferring process is given in Figure 4.2.

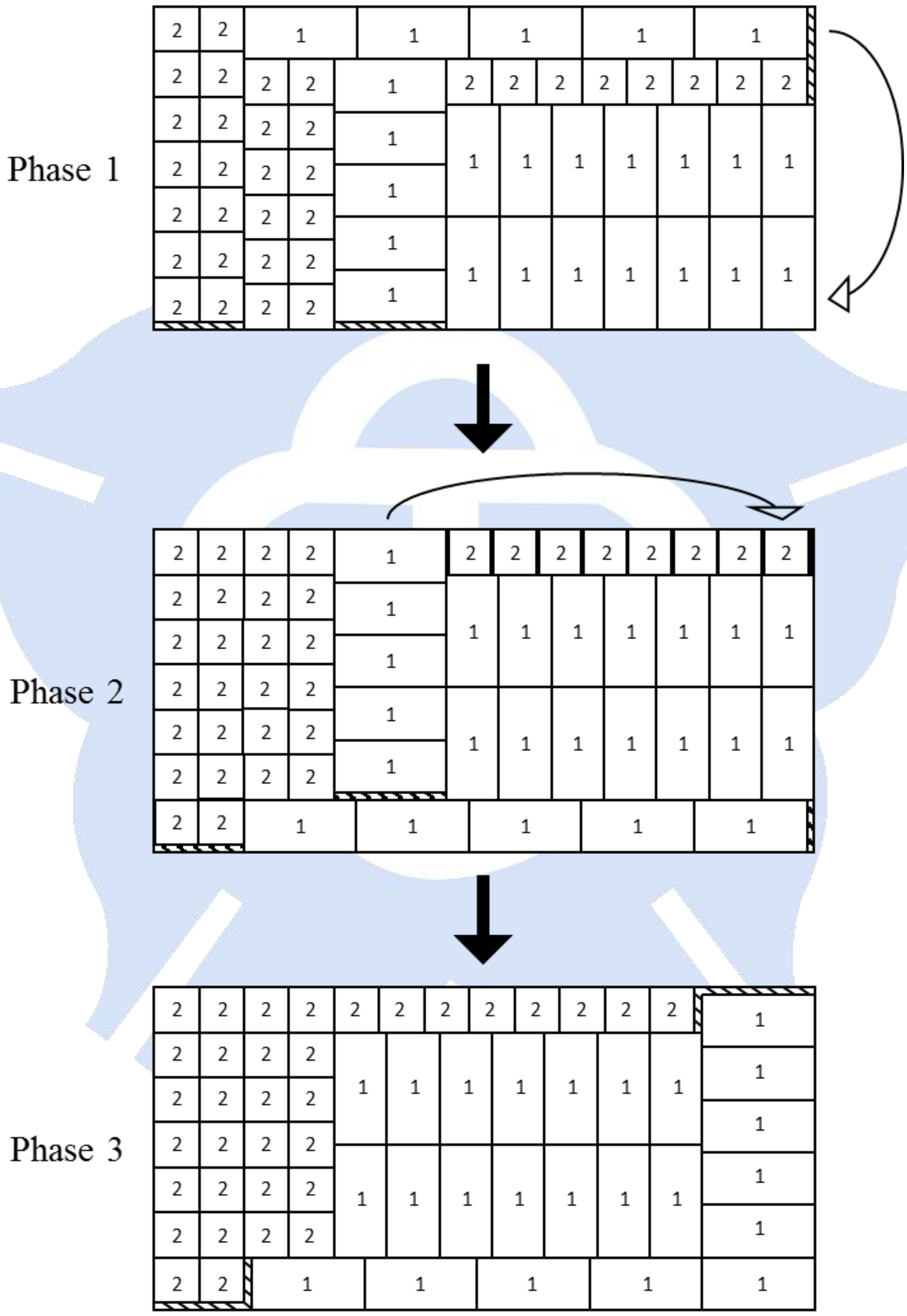


Figure 4.2. Justifying process of normal pattern.

4.2. Application in Printed Circuit Board

In this section, we develop a solution to PCB cutting stock problem and apply previous algorithm to find the optimal pattern.

In PCB industry, customer will order PCB Company a combination of several PCB pieces which is called Shipping Panel and give PCB Company the Sipping Panel size. To manufacture Shipping Panel, PCB Company produces many Shipping Panels at the same time on Working Panel. Company has to find out a best Working Panel size for minimum waste while they cut off the Working Panel from Core Board. Figure 4.3 shows a Shipping Panel, a Working Panel and a Core Board.

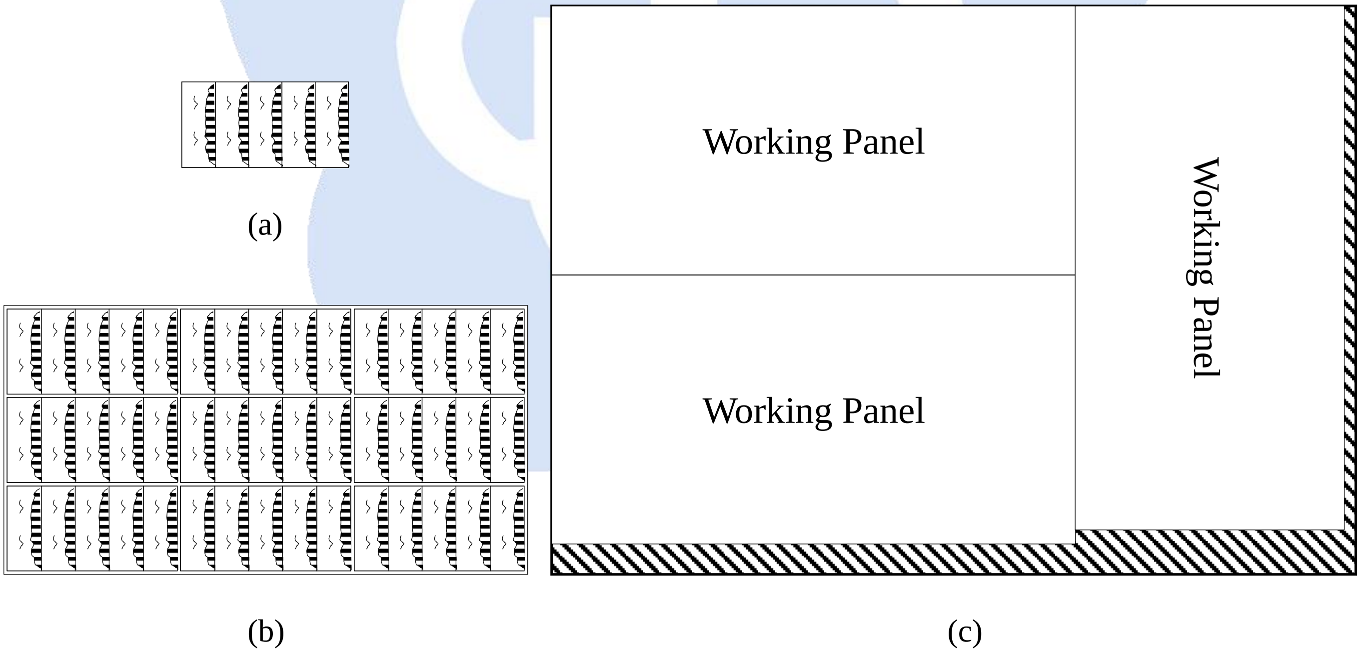


Figure 4.3. (a) A Shipping Panel with five PCB pieces, (b) a Working Panel with nine Shipping Panels and (c) a Core Board which can be cut to three Working Panels and the waste of trim loss on the slash area.

There are some assumptions while cutting off the Shipping Panel from Working Panel and Core Board:

1. Shipping Panel size $l \times w$ is given by customer. See Figure 4.4(a).
2. Only guillotine cut is allowed.
3. The orientation of Shipping Panel is fixed, which means a Shipping Panel of length x and width y ($x \neq y$) is not equal to a Shipping Panel of length y and width x.
4. Cut are made to be infinitely thin, which means the waste of cut line can be ignored.
5. The gap between Shipping Panels and edge of Working Panel need to be preserved 0.118″. See Figure 4.4(b) and(c).
6. On a Working Panel, only two possible directions of Shipping Panel are allowed. They are direction Shipping Panel width parallel to Y-direction and direction Shipping Panel width parallel to X-direction, see Figure 4.4(b) and (c) respectively.

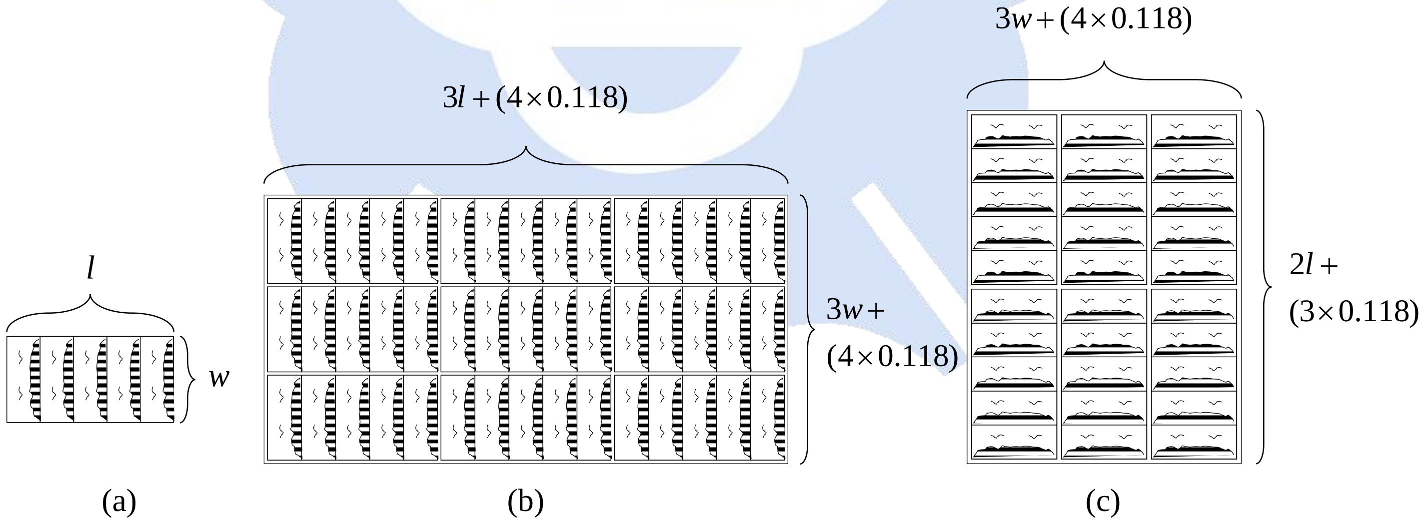


Figure 4.4. (a) A Shipping Panel with size $l \times w$, (b) a Working Panel which its Shipping Panel width parallel to Y-direction and (c) a Working Panel which its Shipping Panel width parallel to X-direction.

To Determine Working Panel size to produce Shipping Panel, formulas for finding possible Working Panel sizes are described as bellow:

When the Shipping Panel width parallel to Y-direction on Working Panel:

Working Panel X-direction length × Working Panel Y-direction length =

$$[al + (a + 1) \times 0.118] \times [bw + (b + 1) \times 0.118] \tag{6}$$

When the Shipping Panel width parallel to X-direction on Working Panel:

Working Panel X-direction length × Working Panel Y-direction length =

$$[aw + (a + 1) \times 0.118] \times [bl + (b + 1) \times 0.118] \tag{7}$$

In Formula (6) and (7), a and b are non-negative integers which represent the number of X-direction Shipping Panel pieces and the number of Y-direction Shipping Panel pieces.

The Working Panel have to meet the restriction that the maximum size cannot greater than size 24.5" × 21.5" and the minimum size cannot less than size 16.2" × 13". The restrictions are shown in the following inequalities:

When the Shipping Panel width parallel to Y-direction on Working Panel:

$$16.2 < [al + (a + 1) \times 0.118] < 24.5 \tag{8}$$

and

$$13 < [bw + (b + 1) \times 0.118] < 21.5 \tag{9}$$

When the Shipping Panel width parallel to X-direction on Working Panel:

$$16.2 < [aw + (a + 1) \times 0.118] < 24.5 \tag{10}$$

And

$$13 < [bl + (b + 1) \times 0.118] < 21.5 \tag{11}$$

For direction Shipping Panel width parallel to Y-direction on Working Panel, we let any possible $[al + (a+1) \times 0.118]$ in Formula (8) be elements x_i in the set $L_Y = \{x_1, .., x_{M_Y}\}$ when $x_1 = \min\{x_i | 16.2 < x_i < 24.5, i \in 1, 2, .., M_Y\}$ and $x_{i+1} > x_i$ for $1 \leq i \leq M_Y$; let any possible $[bw + (b+1) \times 0.118]$ in Formula (9) be elements y_j in the set $W_Y = \{y_1, .., y_{N_Y}\}$ when $y_1 = \min\{y_j | 13 < y_j < 21.5, j \in 1, 2, .., N_Y\}$ and $y_{j+1} > y_j$ for $1 \leq j \leq N_Y$.

For direction Shipping Panel width parallel to X-direction on Working Panel, we let any possible $[aw + (a+1) \times 0.118]$ in Formula (10) be elements x_k in the set $L_X = \{x_1, .., x_{M_X}\}$ when $x_1 = \min\{x_k | 16.2 < x_k < 24.5, k \in 1, 2, .., M_X\}$ and $x_{k+1} > x_k$ for $1 \leq k \leq M_X$; let any possible $[bl + (b+1) \times 0.118]$ in Formula (11) be elements y_h in the set $W_X = \{y_1, .., y_{N_X}\}$ when $y_1 = \min\{y_h | 13 < y_h < 21.5, h \in 1, 2, .., N_X\}$ and $y_{h+1} > y_h$ for $1 \leq h \leq N_X$.

We try every possible combinations of Working Panel size $x_i \times y_j$ and $x_k \times y_h$ which comes from Formula (8) & (9) and Formula (10) & (11) respectively in stock rectangle size 49" × 43", 49" × 41" and 49" × 37" by Algorithm for Multiple-Size to find the optimal utilization Working Panel size (Cui's Algorithm A can solve this problem as well because it is a CSPER). The Algorithm for PCB cutting stock problem is shown as below:

Algorithm for PCB Cutting Stock Problem

Step 1. Let $L = 49, W = y = 37; U = 0$.

Step 2. Let $i = 1, j = 1; k = 1, h = 1;$

Step 3. Call Algorithm for Multiple-Size.

Step 4. Let stock piece $l_1 = x_i, w_1 = y_j$. Run Algorithm for Multiple-Size.

Step 5. If objective value $U < F(49, y)$, let $U < F(49, y)$, let $OptSize = x_i \times y_j$ and $OptCoreboard = 49 \times y$. Let $j = j + 1$. If $j > N_Y$, go to step 6. Otherwise, go to step 4.

Step 6. Let $j = 1$ and $i = i + 1$. If $i > M_Y$, go to step 7 and remove memory L_Y and W_Y. Otherwise, go to step 4.

Step 7. Let stock piece $l_1 = x_k, w_1 = y_h$. Run Algorithm for Multiple Size.

Step 8. If objective value $U < F(49, y)$, let $U = F(49, y)$, let $OptSize = x_k \times y_h$ and $OptCoreboard = 49 \times y$. Let $h = h + 1$. If $h > N_X$, go to step 9. Otherwise, go to step 7.

Step 9. Let $h = 1$ and $k = k + 1$. If $k = M_X$, go to step 10. Otherwise, go to step 7.

Step 10. If $y < 41$, let $y = 41$ and go to step 2. If $y = 41$, let $y = 43$ and go to step2. Otherwise, go to step 11.

Step 11. Objective value U, $OptSize$ and $OptCoreboard$ have been found.

Chapter 5 Conclusions and Future Research

In this study we provide an algorithm for finding normal pattern and simplest optimal pattern in multiple-size cutting stock problem. The objective is to minimize the waste of trim loss while cutting different sizes of stock piece off the stock rectangle. The algorithm includes 4 stages: 1. Normal Length Generating, 2. Dynamic Programming Algorithm Running, 3. Section Back Tracking and 4. Normal Pattern Justifying. Based on clear recursive function in dynamic programming algorithm (Algorithm for Multiple-Size), the calculating procedure is easy to design and understand, which also increase the possibility to apply in practical usage.

For further studies extended from our thesis, several recommendations and directions are summarized as follows:

1. Considering the production proportion between different sizes of stock piece while seeking for the simplest optimal pattern and normal pattern.
2. Improving the efficiency of dynamic programming calculation such as reducing the number of unnecessary step off points.
3. Considering working panel production capacity to trade the production cost off against the waste of trim loss in PCB cutting stock problem.

Appendix A. A Case in Formula (2)

Suppose that we have two sizes of stock piece need to be cut from a stock rectangle. The sizes are described as below:

Stock Rectangle: $L = 8, W = 7$

Stock Piece: $l_1 = 5, w_1 = 2; l_2 = 4, w_2 = 3$

We've already known that this size would have full pattern:

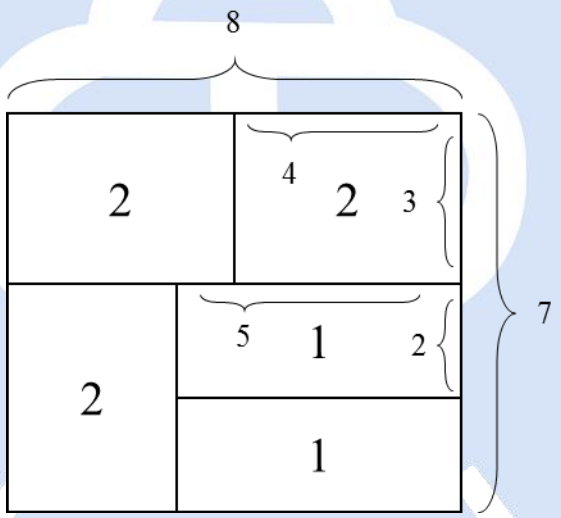


Figure A.1. Full pattern of rectangle $x \times y = 8 \times 7$.

Now we apply Formula (2) to find the optimal pattern, the results of two possible cuts with two sizes of stock piece while $F(x,y) = F(8,7)$ are shown in the following figure:

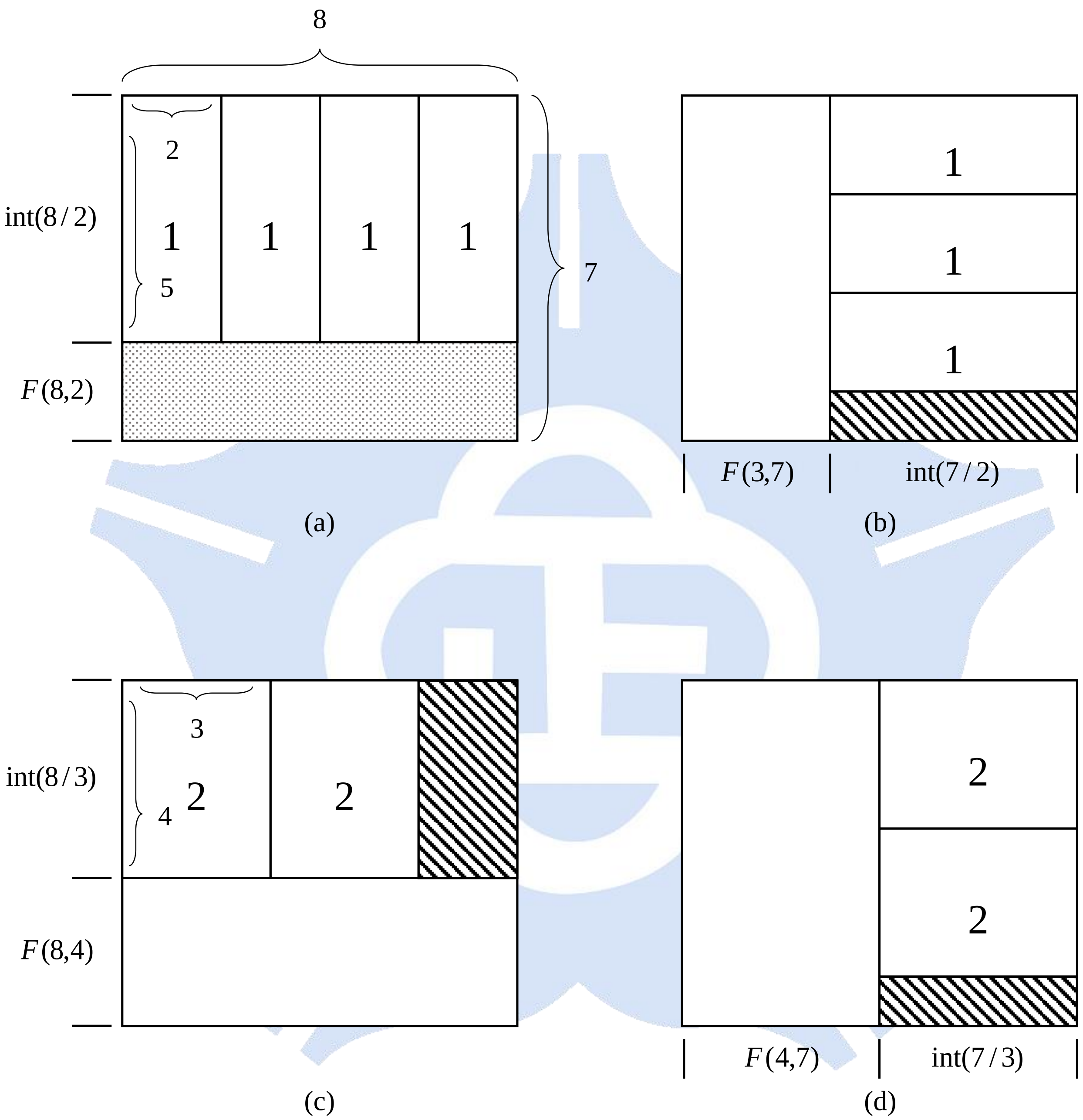


Figure A.2. Two possible cuts with two sizes of stock piece on rectangle $x \times y = 8 \times 7$. The value $int()$ must multiply $l_i w_i$, we ignored it for neat purpose.

The cut in every stage of dynamic programming must have no waste if we want to find the optimal pattern of this case (full pattern). Only Figure A.2(a) meets this condition after putting these two sizes of stock piece into the stock rectangle $x \times y = 8 \times 7$. Now let's trace the value $F(x, y)$ on rectangle $x \times y = 8 \times 2$ (grey area in Figure A.2(a)):

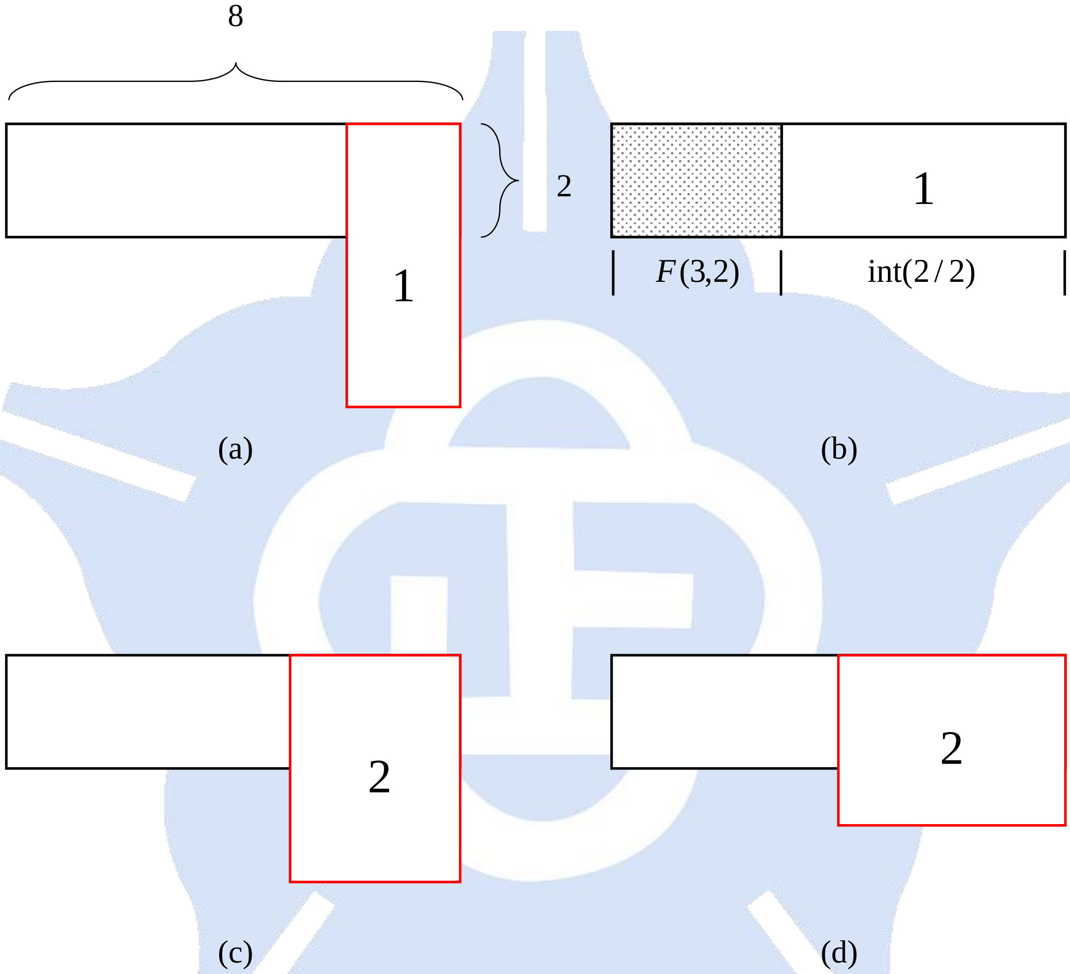


Figure A.3. Two possible cuts with two sizes of stock piece on rectangle $x \times y = 8 \times 2$. The value $int()$ must multiply $l_i w_i$, we ignored it for neat purpose.

In Figure A.3, the only possible cut is (b). However, the rest of stock rectangle $x \times y = 3 \times 2$ (grey area in Figure A.3(b)) cannot be put any stock piece inside itself, proving that Formula (2) is infeasible to find the optimal pattern in multiple-size CSP.

Appendix B. A Case in Formula (3)

We extend the example of Appendix A. in this section and give an application of Formula (3). The results of four possible cuts with two sizes of stock piece while $F(x,y) = F(8,7)$ are shown in the following figure:

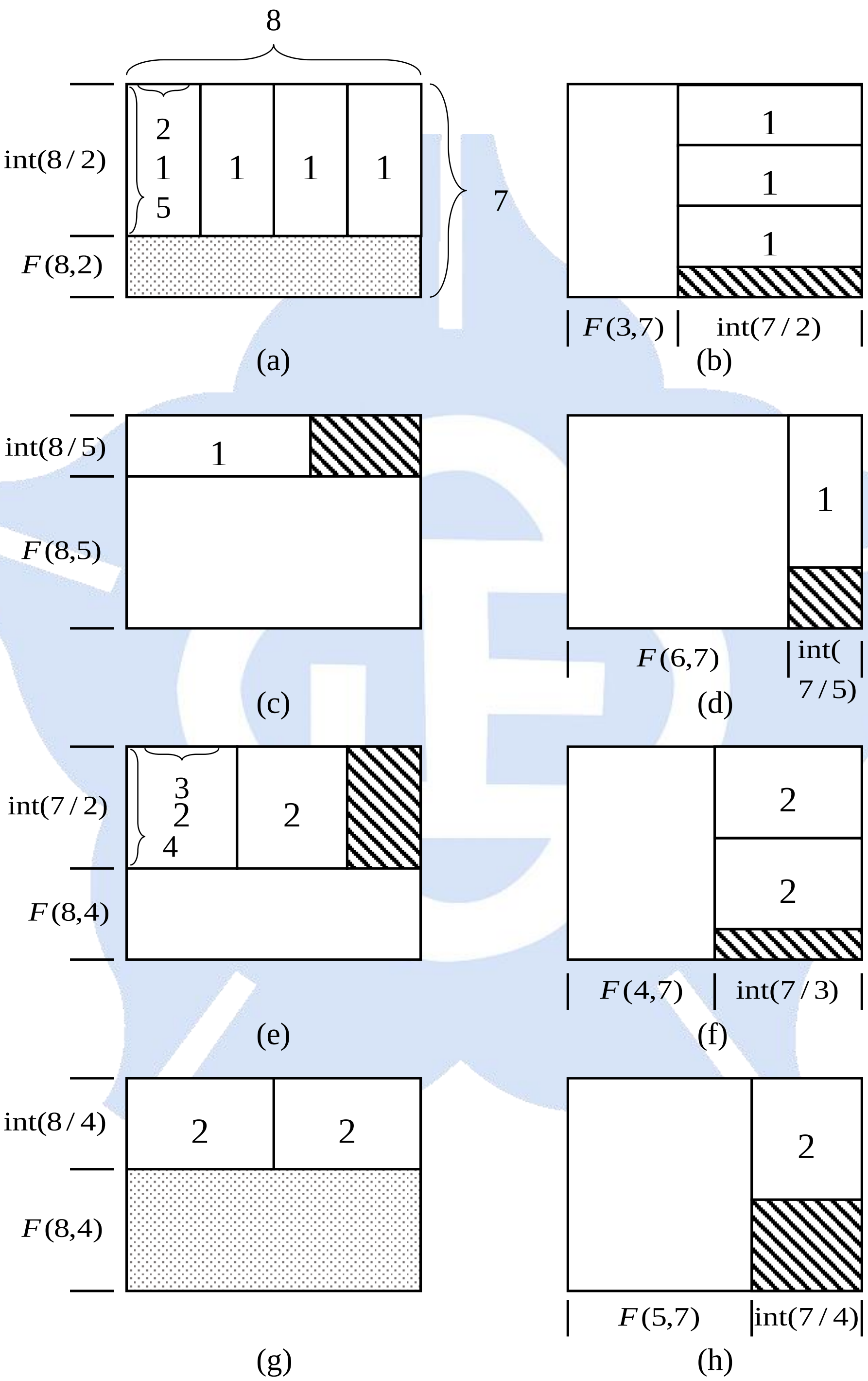


Figure B.1. Four possible cuts with two sizes of stock piece on rectangle $x \times y = 8 \times 7$. The value $int()$ must multiply $l_i w_i$, we ignored it for neat purpose.

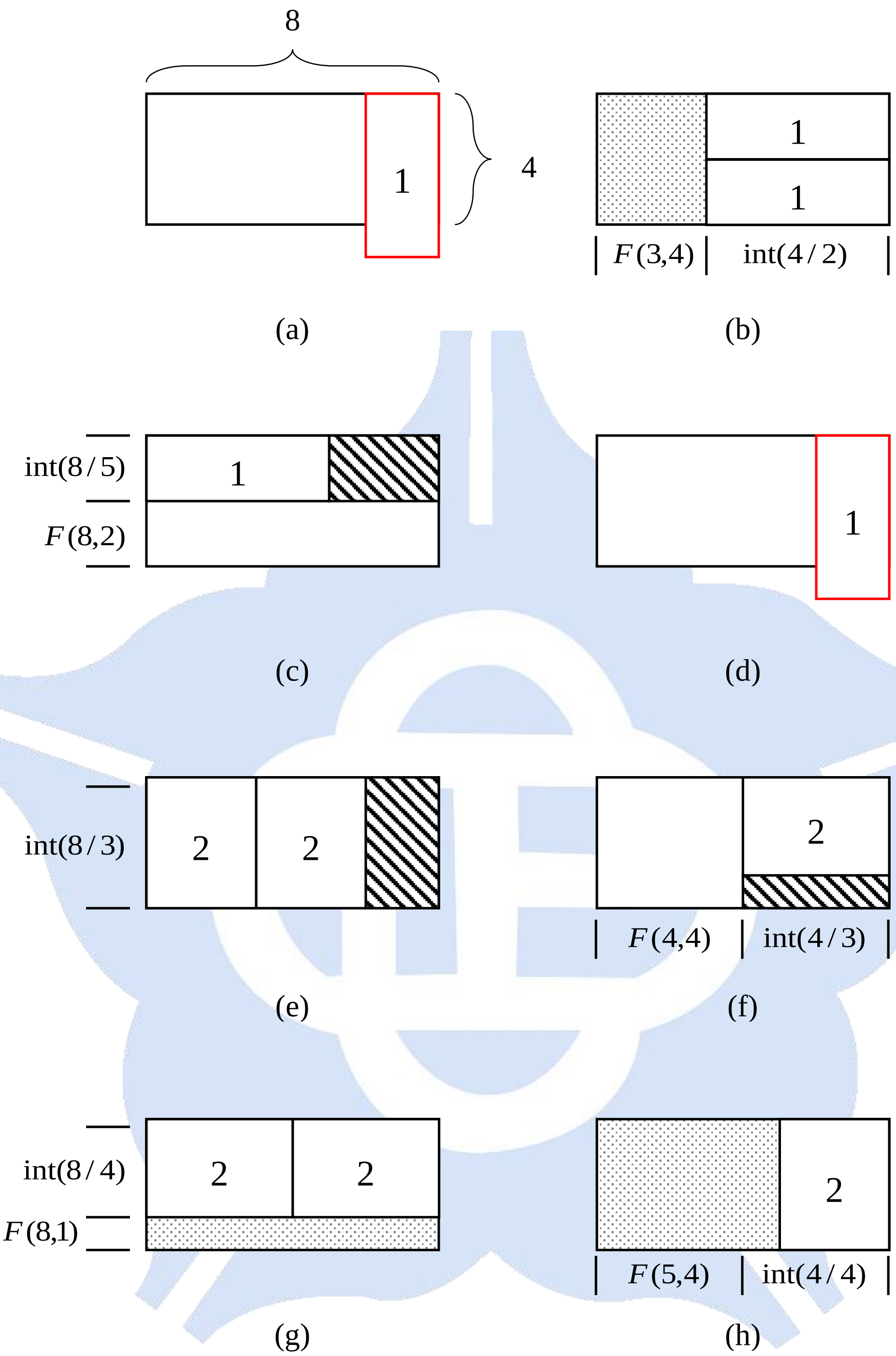


Figure B.2. Four possible cuts with two sizes of stock piece on rectangle $x \times y = 8 \times 4$. The value $int()$ must multiply $l_i w_i$, we ignored it for neat purpose.

In Figure B.1, only (a) and (g) have no waste after putting these two stock piece sizes into the stock rectangle $x \times y = 8 \times 7$. It's trivial that $F(8,2)$ can only be cut to stock piece $l_1 \times w_1 = 5 \times 2$ with one piece and the rest of stock rectangle $x \times y = 3 \times 2$ become a waste. Thus, we just need to trace the grey area in Figure B.1(g) on rectangle $x \times y = 8 \times 4$, as shown in Figure B.2. In Figure B.2, there are three cuts meet no waste: Figure B.2(b), (g) and (h). It's trivial that Figure B.2(g) $F(8,1) = 0$ because it cannot be cut to any stock piece. Figure B.2(b) $F(3,4)$ is exactly the same as stock piece $l_2 \times w_2 = 4 \times 3$ and Figure B.2(h) $F(5,4)$ can be cut to two stock pieces $l_1 \times w_1 = 5 \times 2$, both of which reach the full pattern respectively. Thus, Formula (3) can guarantees the optimal pattern in multiple-size CSP.

References

Agrawal, P. (1993). Minimising trim loss in cutting rectangular blanks of a single size from a rectangular sheet using orthogonal guillotine cuts. *European Journal of Operational Research, 64*(3), 410-422.

Alvarez-Valdés, R., & Parajón, A. (2002). A tabu search algorithm for large-scale guillotine (un) constrained two-dimensional cutting problems. *Computers & Operations Research, 29*(7), 925-947.

Beasley, J. (1985a). Algorithms for unconstrained two-dimensional guillotine cutting. *Journal of the Operational Research Society, 36*(4), 297-306.

Beasley, J. (1985b). An exact two-dimensional non-guillotine cutting tree search procedure. *Operations Research, 33*(1), 49-64.

Bellman, R. (1952). On the theory of dynamic programming. *Proceedings of the National Academy of Sciences, 38*(8), 716-719.

Belov, G., & Scheithauer, G. (2006). A branch-and-cut-and-price algorithm for one-dimensional stock cutting and two-dimensional two-stage cutting. *European journal of operational research, 171*(1), 85-106.

Chen, C.-L. S., Hart, S. M., & Tham, W. M. (1996). A simulated annealing heuristic for the one-dimensional cutting stock problem. *European journal of operational research, 93*(3), 522-535.

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

Cintra, G., Miyazawa, F. K., Wakabayashi, Y., & Xavier, E. (2008). 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.

Cui, Y. (2005). Dynamic programming algorithms for the optimal cutting of equal rectangles. *Applied Mathematical Modelling, 29*(11), 1040-1053.

Cui, Y., & Zhou, R. (2002). Generating optimal cutting patterns for rectangular blanks of a single size. *Journal of the Operational Research Society, 53*(12), 1338-1346.

De Carvalho, J. V. (2002). LP models for bin packing and cutting stock problems. *European Journal of Operational Research, 141*(2), 253-273.

Dyckhoff, H. (1981). A new linear programming approach to the cutting stock problem. *Operations Research, 29*(6), 1092-1104.

Dyckhoff, H. (1990). A typology of cutting and packing problems. *European Journal of Operational Research, 44*(2), 145-159.

Farley, A. (1992). Limiting the number of each piece in two-dimensional cutting stock patterns. *Mathematical and computer modelling, 16*(1), 75-87.

Fayard, D., & Zissimopoulos, V. (1995). An approximation algorithm for solving unconstrained two-dimensional knapsack problems. *European Journal of Operational Research, 84*(3), 618-632.

Furini, F., & Malaguti, E. (2013). Models for the two-dimensional two-stage cutting stock problem with multiple stock size. *Computers & Operations Research, 40*(8), 1953-1962.

Gilmore, P., & Gomory, R. E. (1965). Multistage cutting stock problems of two and more dimensions. *Operations research, 13*(1), 94-120.

Gilmore, P., & Gomory, R. E. (1966). The theory and computation of knapsack functions. *Operations Research, 14*(6), 1045-1074.

Gilmore, P. C., & Gomory, R. E. (1961). A linear programming approach to the cutting-stock problem. *Operations research, 9*(6), 849-859.

Haessler, R. W. (1980). A note on computational modifications to the Gilmore-

Gomory cutting stock algorithm. *Operations Research*, *28*(4), 1001-1005.

Herz, J. (1972). Recursive computational procedure for two-dimensional stock cutting. *IBM Journal of Research and Development, 16*(5), 462-469.

Hifi, M. (1997). The DH/KD algorithm: a hybrid approach for unconstrained two-dimensional cutting problems. *European Journal of Operational Research, 97*(1), 41-52.

Hifi, M., & Roucairol, C. (2001). Approximate and exact algorithms for constrained (un) weighted two-dimensional two-staged cutting stock problems. *Journal of combinatorial optimization, 5*(4), 465-494.

Leung, T., Yung, C., & Troutt, M. D. (2001). Applications of genetic search and simulated annealing to the two-dimensional non-guillotine cutting stock problem. *Computers & industrial engineering, 40*(3), 201-214.

Levine, J., & Ducatelle, F. (2004). Ant colony optimization and local search for bin packing and cutting stock problems. *Journal of the Operational Research Society, 55*(7), 705-716.

Morabito, R., Arenales, M., & Arcaro, V. (1992). An and—or-graph approach for two-dimensional cutting problems. *European Journal of Operational Research, 58*(2), 263-271.

Shen, X., Li, Y., Yang, J., & Yu, L. (2007). *A Heuristic particle swarm optimization for cutting stock problem based on cutting pattern.* Paper presented at the International Conference on Computational Science.

Wäscher, G., Haußner, H., & Schumann, H. (2007). An improved typology of cutting and packing problems. *European journal of operational research, 183*(3), 1109-1130.

Zak, E. J. (2002). Row and column generation technique for a multistage cutting stock problem. *Computers & Operations Research, 29*(9), 1143-1156.