國 立 交 通 大 學

機 械 工 程 學 系

博 士 論 文

二次規劃法配合全域策略於多種裁切庫存問題之研究

Sequential Quadratic Programming Method with Global Strategy

for Multiple Types of Multi-polygon Cutting-stock Problem

研 究 生：余 明 達

指導教授：洪 景 華 老師

林 聰 穎 老師

曾 錦 煥 老師

中華民國 九十八年七月

二次規劃法配合全域策略於多種裁切庫存問題之研究
Sequential Quadratic Programming Method with Global Strategy for Multiple Types of Multi-polygon Cutting-stock Problem

研 究 生：余明達　　Student： Ming-Ta Yu

指導教授：洪景華　　Advisor：Chinghua Hung

林聰穎　　Tsung-Yin Lin

曾錦煥　　Ching-Huan Tseng

國 立 交 通 大 學

機 械 工 程 學 系

博 士 論 文

A Dissertation

Submitted to Department of Mechanical Engineering

College of Engineering

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

in

Mechanical Engineering

July 2009

Hsinchu, Taiwan, Republic of China

中華民國九十八年七月

二次規劃法配合全域策略於多種裁切庫存問題之研究

研究生：余明達　　指導教授：洪景華
林聰穎
曾錦煥

國立交通大學機械工程學系

摘要

裁切庫存問題是一種有限制條件的最佳化問題，它在討論如何將所欲加工零件的外型排列在材料中，能提高材料的利用率，且沒有重疊。

裁切庫存問題存在於許多的工業中，例如紡織業、成衣業、紙張製造業、造船業及板金業。裁切庫存問題可分為許多類型。例如：方形物件的排列、不規則物件的排列、方形材料的使用、不規則材料的使用、單一材料與多種材料等。本研究將重點集中在不規則物件的排列上，將裁切庫存問題規劃成限制最佳化問題的形式，並利用求解限制最佳化問題常用的序列二次規劃法配合本研究提出的全域搜尋策略，來找到良好的解。並利用虛擬物件的概念將不規則材料問題與多材料問題簡化成單一方型材料問題，使得所提出的搜尋策略能廣泛地應用到多種不同的問題。此外，本研究還提出一種適用於序列二次規劃法的物件重疊的指標，與一簡化的模型以達到簡化限制條件計算的目的。

Sequential Quadratic Programming Method with Global Strategy for Multiple Types of Multi-polygon Cutting-stock Problem

Student: Ming-Ta Yu

Advisors: Dr. ChingHua Hung
Dr. Tsung-Yin Lin
Dr. Ching-Huan Tseng

Department of Mechanical Engineering
National Chiao Tung University

ABSTRACT

The cutting-stock problem, which considers how to arrange the component profiles on the material without overlaps, can increase the utility rate of the stock, and is thus a standard constrained optimization problem.

The cutting-stock problem is relevant in many industries, such as textile, garment, paper, ship building, and sheet metal industries. The cutting-stock problem can be classified in many types, such as: rectangle object problem, irregular object problem, rectangle stock problem, irregular stock problem, single-stock problem, and multi-stock problem. This study focuses on the irregular object problem, and formulates it as a standard constrained optimization problem. The Sequential Quadratic Programming method, which is famous for solving a constrained optimization problem, is used with the global strategies, which are proposed in this study, for obtaining a good solution. This

study also proposes a virtual object strategy to simplify the irregular stock problem and the multi-stock problem as a single rectangular stock problem. Additionally, this study proposes an overlap index, which is suitable for the Sequential Quadratic Programming method, and proposes a simplification model for simplifying the calculation of constraints.

誌謝

從交大大學部、碩士班、博士班，待在交大的日子已經十二年了。在交大的日子佔了到目前為止生命快一半的時光。這段日子裡要感謝的人很多，最要感謝的，就是恩師曾錦煥老師，曾老師時時引導我，改變我的觀念，鼓勵我，激發我的鬥志。在老師走了之後，洪景華老師跟林聰穎學長不辭勞苦地接下了指導我的工作，讓我能夠繼續從事最佳化的研究，這裡也要向老師及學長致上深深的感謝。

感謝我的口試委員，蔡忠杓老師、宋震國老師、徐瑞坤老師以及陳申岳總經理。謝謝您們給予學生的建議以及指導。謝謝實驗室的師兄弟，謝謝您們和我討論專業上的問題，也聽我論文做不出來的抱怨。更懷念那些大家一起睡在實驗室的日子。尤其感謝嘉宏，沒有您我就沒有機緣進這個實驗室。特別感謝陸聯公司的長官以及同事，在我論文最忙時，多替我分擔了許多工作上的負荷。

感謝我的父母以及家人對我的鼓勵，在我投稿沒有回音時，給我的鼓勵、陪伴、與支持。感謝我的愛人，在我壓力大時，陪伴我、帶我到處走走，紓解我的壓力；在我不開心時，逗我笑。

感謝大家。

CONTENTS

CONTENTS OF TABLES

CONTENTS OF FIGURES

NOTATIONS

A	a matrix used to consider constraints in QP sub-problem
a	the object index which will be swapped
B	a matrix used in the solution of the normalized QP sub-problem
b	a vector used to consider constraints in QP sub-problem
b	the object index which will be swapped
c_0	a variable used to normalize the QP sub-problem
c_i	a variable used to normalize the QP sub-problem
D	the solution of the SQP method
D	a depth
$\mathbf{D_{best}}$	the best solution of the whole solving process
D_{Mjk}	the maximum depth between object j and object k
$\mathbf{D_{sub}}$	the solution in the swapping sub-process
$\mathbf{D_{subbest}}$	the best solution in the swapping sub-process
d	the vector form of design variables
E	a matrix used to update the approximate Hessian matrix
F_{best}	the cost function value of the best solution in the whole solving process
F_{sub}	the cost function value of the solution in the swapping sub-process
$F_{subbest}$	the cost function value of the best solution in the swapping sub-process

$\mathbf{g}$	the vector form of constraints
$\mathbf{g}_a$	the vector form of the active-set constraints
g1	the vector form of overlap constraints between objects
g2	the vector form of x-lower boundary constraints
g3	the vector form of y-lower boundary constraints
g4	the vector form of y-upper boundary constraints
g5	the vector form of overlap constraints between object and virtual object
g1′	the calculation simplification form of **g1**
g5′	the calculation simplification form of **g5**
H	the approximate Hessian matrix
IterNo	the variable indicates the iteration number of the whole solving process
IterMax	the maximum iteration number of the whole solving process
$k_{\max}$	the maximum iteration number of the SQP method
M	a matrix used to consider the constraints in normalized QP sub-problem
N	the number of design variables
N_v	the number of virtual objects
O_i	the reference point of object i
P	a matrix used to update the approximate Hessian matrix
p'	the number of the active-set constraints

q	a variable used to calculate the step size
R_k	a variable used to calculate the step size of iteration k
s	the vector form of design variables in the normalized QP sub-problem
SwapNo	the variable indicates the number of swap action in one iteration
SwapMax	the maximum allowable swapping number
T	the temperature of the Simulated Annealing Algorithm
U	an upper triangular matrix for normalizing the QP sub-problem
u	a vector used to update the approximate Hessian matrix
v	a vector used in the solution of the normalized QP sub-problem
w	a vector used to update the approximate Hessian matrix
x_i	the x-coordinate value of the reference point of object i
x_{li}	the lower boundary of object i in the x-direction
x_{ui}	the upper boundary of object i in the x-direction
y_i	the y-coordinate value of the reference point of object i
y_{li}	the lower boundary of object i in the y-direction
y_{ui}	the upper boundary of object i in the y-direction
y_{width}	the width of the stock
z	a vector used to update the approximate Hessian matrix
α_k	the step size of the SQP method

$\mathbf{\Delta}$	a vector used to consider the constraints in normalized QP sub-problem
$\mathbf{\Delta d}$	the change in design variables
ΔE	the cost function increment of the Simulated Annealing method
ε	a constant used to check the convergence
ε_c	a constant used to check the constraint violation is serious or not
θ_i	the orientation of object i
$\mathbf{\Lambda_0}$	a matrix used to consider the cost function in normalized QP sub-problem
$\boldsymbol{\mu}$	the vector form of the Lagrange multipliers
σ	the random number of the Simulated Annealing method
Ψ	a variable used to update the approximate Hessian matrix
∇f	the gradient of the cost function

CHAPTER 1 INTRODUCTION

The cutting-stock problem is the problem considers how to arrange the components of products in the stock that will enhance the stock utility or reduce the necessary stock. This problem is often also called the “packing problem” or “nesting problem.” These three names may be used for different types of problems considering how to arrange components in the stock that can enhance the stock utility. However, they are all called the cutting-stock problem in this study.

The main purpose of this problem is reducing the material cost. The cutting-stock problem is a key consideration in many manufacturing industries, such as textile, garment, metalware, paper, ship building, and sheet metal industries. In these industries, the products or the components will be cut from the stock. According to the statistics in by DGBAS (1998) and by DGBAS (2003), the cost of stock is about 50% in whole expense of manufacturing industries in Taiwan. It means that enhancing the stock utility will be helpful for reducing the outgoing of companies in these industries.

Because characteristics of the cutting-stock problem in different applications may be different, the cutting-stock problem can be classified in different types by its characteristic in the next section. The classification will be helpful to know how many types should be solved. The nature of the cutting-stock problem will also be introduced in this chapter to know how to solve the problem. After knowing the

classification and the nature of this problem, the problem is understood well and the objectives of this study can be obtained.

1.1 Classification of the Cutting-stock Problem

The cutting-stock problem comprises the stock and the object. Therefore, the cutting-stock problem can be classified by the characteristics of the stock and the object. The cutting-stock problem may be one-, two-, or three-dimensional problem. The applications of the one-dimensional problem are cutting pipes, steel bars, and etc. Different length pipes should be cut from many long pipes, and how to cut the pipes may use the least amount of long pipes is a topic of the one-dimensional cutting-stock problem. The applications of the three-dimensional problems are packing boxes into a container, arranging components in the working space in the rapid prototyping industry, and etc. Even there are one- and three-dimensional cutting-stock problem, the two-dimensional cutting-stock problem is the most popular one. Therefore, this study focuses on the two-dimensional cutting-stock problem.

In the two-dimensional problem, the shape of the objects can be classified in rectangle and irregular shapes. Rectangle is a special shape because rectangles can be arranged very close even there are different kinds of rectangles arranged together. It is because every angle of a rectangle is 90 degree. With the same reason, rectangles are easy to be represented with heights and widths. However, the profile of product in real

applications is usually not rectangle. To represent the irregular object and to check the overlap between objects are difficult. An irregular object may be a polygon or may have some curves. In general, any curve of an object is usually approximated by some edges with acceptable tolerance. The approximation method is used by Nee, *et al.*, (1986), Koroupi and Loftus (1991), and Cheng and Rao (1999). Kershner (1968) showed that no convex polygon with more than six sides can be arranged without gaps, and there are only eight kinds of pentagons and three kinds of hexagons that can be arranged without gaps. For arranging these polygons without gaps, there should be only one kind polygon in the problem as shown in Fig. 1-1, i.e. different kind of these polygons cannot be arranged together without gaps. This study focuses on the cutting-stock problem with irregular object because the cutting-stock problem with irregular objects is more complex and difficult than the problem only arranging rectangles.

According to how many kinds of objects, the cutting-stock problem can be divided into single-polygon problem and multi-polygon problem. There is only one kind of object in the single-polygon problem, as shown in Fig. 1-2(a). The single-polygon problem is the type of mass production. In this problem, the relative position between objects is obtained firstly and arranged repeatedly. If there are many kinds of objects, it is called the multi-polygon problem, as shown in Fig. 1-2(b). In

this kind of problem, there is not a regular relationship between object positions. Every object has to be arranged individually, and the problem is more complex than the single-polygon problem. Therefore, this study focuses on the multi-polygon problem.

There is another kind of problem that combines the single-polygon and multi-polygon problems and it is called the “combinational problem” in this study. A product may have some components and the product may be mass produced. The cutting-stock problem will be very complex if considering every object as an independent individual. To simplify the problem, the components of a product may be arranged as a cluster first, and then the cluster is nested instead of the original components on the stock. This “clustering-then-nesting” strategy is useful for simplifying the combinational problem because the number of objects is reduced. The advantage of the clustering-then-nesting strategy is not only for arranging, but manufacturing. The manufacturing time may decrease because the components of a product can be cut at the same time by using a die. For the manufacture efficiency, a die may be designed for cutting all components at the same time. Thus, all components will be arranged together, and these components can be used as a single object in mass production. Therefore, it combined the characteristics of the single-polygon and multi-polygon problems.

Classifying by the characteristics of stock is similar to classify by objects. The shape of stock may also be rectangular or irregular. The shape of stock is usually rectangle, but in some applications, such as leather industry, the stock may have irregular shapes. In fact, every manufacture industry may have irregular stocks because some remainder stocks of the last cutting can be reused. The remainder stocks usually have irregular profiles. The irregular stock problem is more complex than the rectangular stock problem because of the irregular profile. Both of them will be discussed in this study.

In some applications, a stock with a very long length, such as a steel roll, can contain all objects at the same time. It is called “single-stock problem” in this study. The length of the stock in this problem is considered as infinite because all objects can be arranged in the stock and the necessary length for manufacture is less than the stock length. To arrange objects on a stock with infinite length is a typical problem in all types of the cutting-stock problem. If a stock can not provide enough material for all objects, there will be several stocks and it is called “multi-stock problem” in this study. The stock is usually not large enough when using the remainder stock and there will be several stocks. If the size of an object is big, such as ship-building industry, the objects usually can not be arranged in only one stock. There will be several stocks because only one stock is not large enough. The multi-stock problem is more difficult

than the single-stock problem because of the limited length and the plurality choice of stocks. Both of them will be discussed in this study.

In some applications, the stock may have some flaws and the flaw region can not be used to produce an object. For example, there may be some flaws on the leather because of the illness or hurt of the animal when it is alive. This flaw will also be considered in this study.

As the introduction above, the types of the cutting-stock problem considered in this study can be summarized as follows:

1. The problem with two dimensional irregular objects.
2. The multi-polygon problem.
3. The combinational problem.
4. The rectangular stock problem and the irregular stock problem.
5. The single-stock problem and the multi-stock problem.
6. The flaw stock problem.

The above-mentioned six types of problems are classified by the object or the stock individually but every application both has the object and the stock. Therefore, four cases are discussed in this study and the cases are listed as follows:

1. The combinational problem.
2. The multi-polygon problem with a rectangular stock.

3. The multi-polygon problem with several rectangular stocks. The stocks have the same width.
4. The multi-polygon problem with several stocks. The stocks may be irregular and have different widths. There may be some flaws on the stocks.

1.2 Nature of the Cutting-stock problem

In fact, the cutting-stock problem is a type of constrained optimization problem. A constrained optimization problem is composed of three main parts:

1. Design variable(s)
2. Cost function(s)
3. Constraint(s).

The solving method improves the cost function and also satisfies the constraints via adjusting the design variables. As this formulation, the cost function of the cutting-stock problem is the stock utility. The design variables are the states of the objects, i.e. the profile of the components. For example, an object has three degree-of-freedom, i.e. two translations and one rotation, in the two-dimensional space. Therefore, three design variables, i.e. two position components and one orientation, are needed to describe the state of an object in a stock.

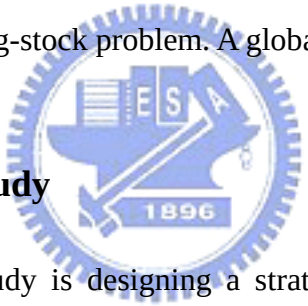
There are four kinds of constraints in the cutting-stock problem, and those constraints are listed as follows:

1. Overlap constraint.
2. Escape constraint.
3. Orientation constraint
4. Gap constraint.

The overlap constraint is that no object can overlap with others. The escape constraint is that no object can be arranged outside the boundary of the stock. The orientation constraint in the cutting-stock problem is that the object can be arranged with only some specific orientations because of the stock property. For instance, the strength of a metal sheet is different in different orientations because of the rolling direction. If the product will be used to support loading, the orientation with higher strength should be chosen. This kind of constraint is the orientation constraint in the cutting-stock problem. The gap constraint is that the gap between objects should be larger than an acceptable value. This constraint usually exists in the sheet metal industry, and it has a standard as shown in Fig. 1-3 and Table 1-1. The gap constraint depends on the thickness of stocks, methods for feeing stock, and the shape of objects.

The technique for solving the constrained optimization problem can be used to solve the cutting-stock problem because the cutting-stock problem is a type of constrained optimization problem. The solving methods for the optimization problem are classified into the indirect method and direct method by Arora (2004). The indirect

method can find the global optimum solution. The cost function and the constraint must be explicit functions of design variables in the indirect method. That is difficult for the cutting-stock problem. The direct method is a repeated process for finding the "search direction" and the "step size," and the explicit function is not necessary. Therefore, the direct method is more suitable than the indirect method for solving the cutting-stock problem. However, the direct method usually only can find the local optimum solution. The cutting-stock problem has large solution space and many complex constraints. The direct method that may be caught by local optimum trap is not good enough for the cutting-stock problem. A global search strategy is necessary.



1.3 Objectives of This Study

The objective of this study is designing a strategy to solve the cutting-stock problem. After knowing the types and the nature of the cutting-stock problem, the objectives of this study can be listed as follows.

1. Propose a method for checking the overlap between irregular objects.
2. Formulate the cutting-stock problem into the equations with a constrained optimization problem form.
3. Propose a global strategy to improve the solution.
4. Propose a method to consider the irregular shape of the stock.
5. Propose a method to consider the plurality of the stock.

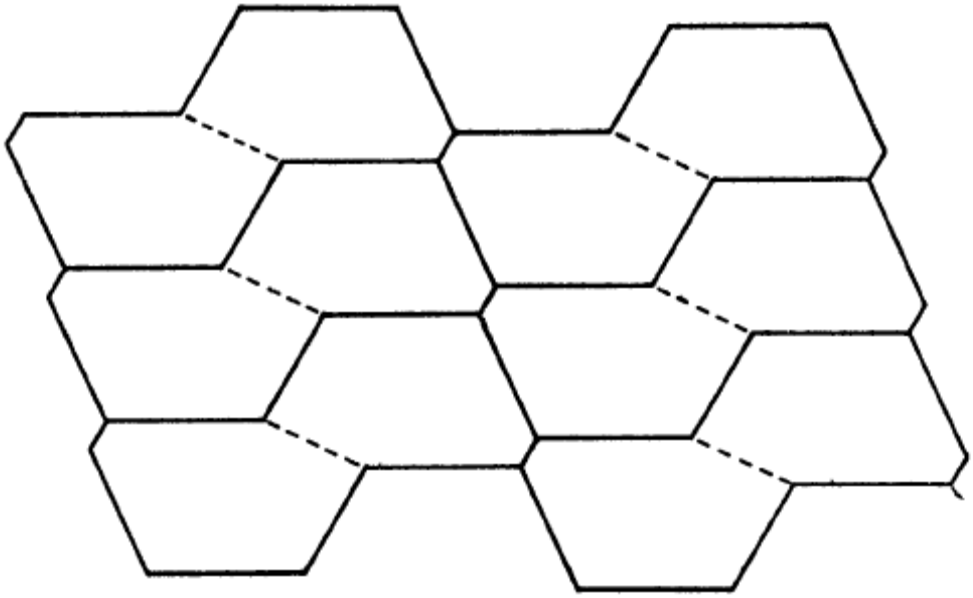
6. Propose a method to consider the flaw on the stock.

1.4 Outlines

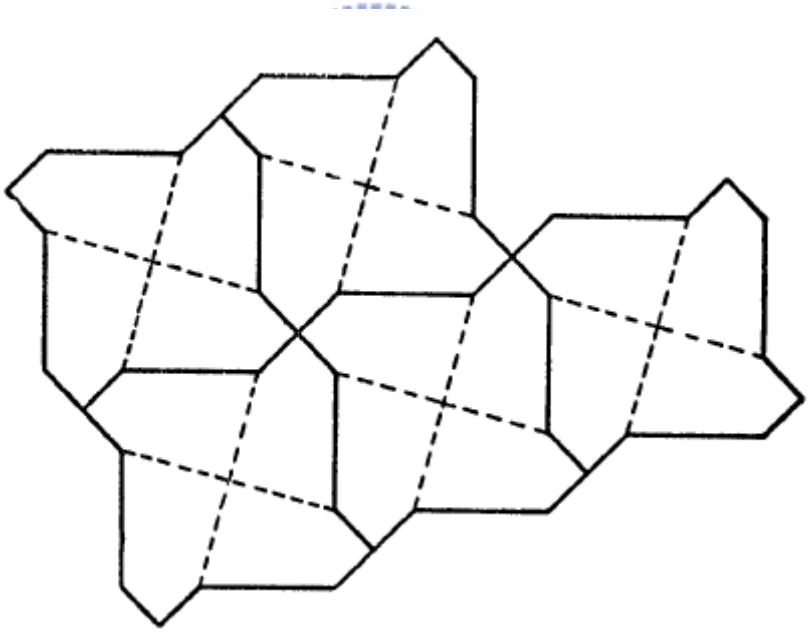
After the introduction of the cutting-stock problem in this chapter, the chapter two will introduce the literatures of the four cases listed in the section 1.1. The chapter three introduces methods used in this study. Some of the methods are proposed in literatures and others are proposed in this study. The chapter four will show the experimental results and some conclusions and future works will be made in the chapter five.

Table 1-1 Types of the gap constraints by Syu (1996).

Thickness (mm)	Feed by hand						Automatic feed	
	Circle		Uncircle		Repeated Shape			
	d	d_1	d	d_1	d	d_1	d	d_1
≦1	1.5	1.5	2	1.5	3	2	3	2
>1~2	2	1.5	2.5	2	3.5	2.5	3	2
>2~3	2.5	2	3	2.5	4	3.5	3	2
>3~4	3	2.5	3.5	3	5	4	4	3
>4~5	4	3	5	4	6	5	5	4
>5~6	5	4	6	5	7	6	6	5
>6~8	6	5	7	6	8	7	7	6
>8	7	6	8	7	9	8	8	7

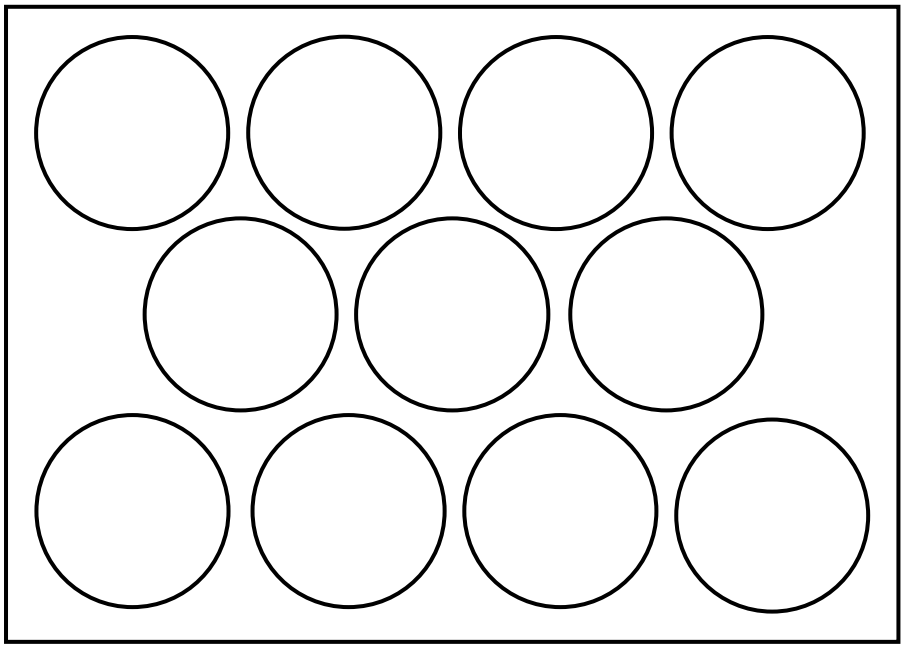


(a) Hexagon

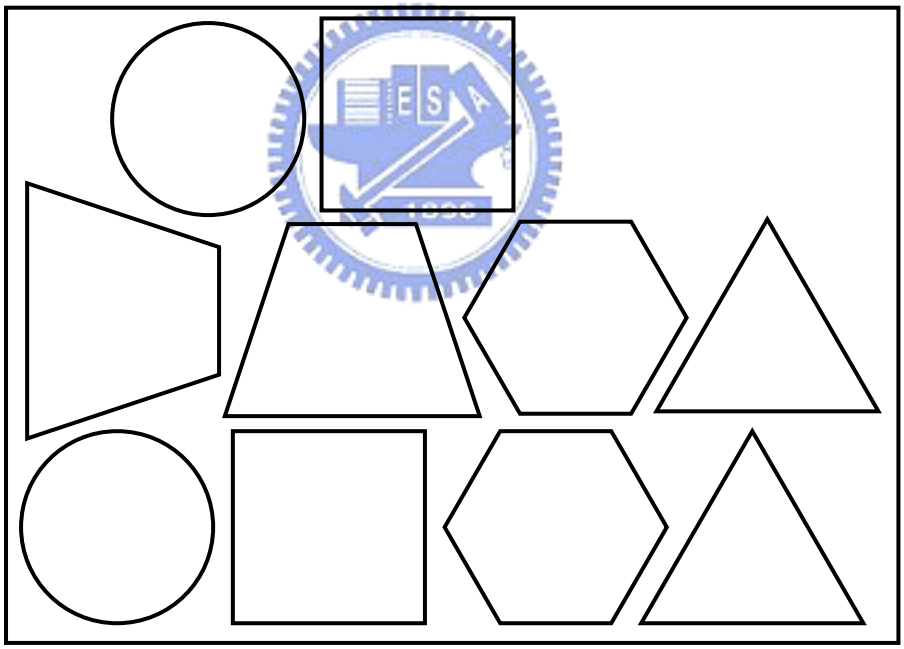


(b) Pentagon

Fig. 1-1 Special polygons with no gap nesting by Kershner (1968).

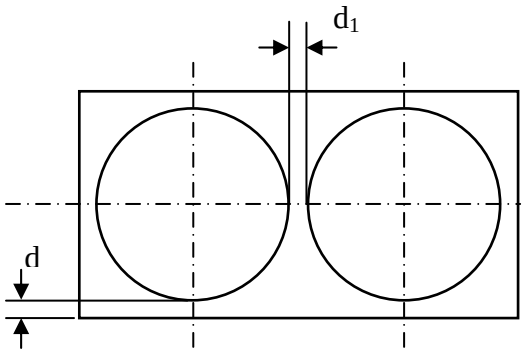


(a) Single-polygon nesting problem

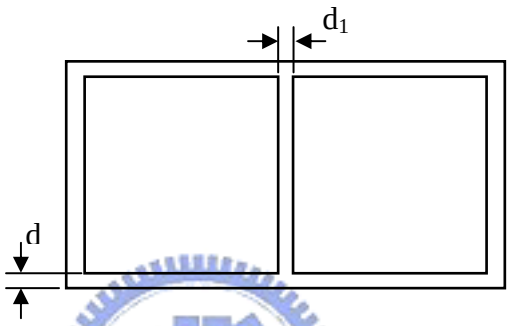


(b) Multi-polygon nesting problem

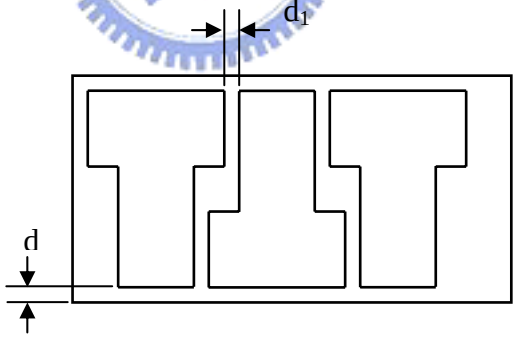
Fig. 1-2 The classifications of the nesting problem.



(a) Circles



(b) Uncircles



(c) Repeated Shapes

Fig. 1-3 Types of the gap constraint by Syu (1996).

CHAPTER 2 LITERATURE REVIEW

The literatures introduced in this chapter are all about multi-polygon. The literatures are divided into four groups that associate with the four cases considered in this study. The methods used in every literature may have three parts: graph treatment, arranging strategy, and search strategy. The graph treatment technique is almost equivalent to the overlap consideration technique because the overlap constraint is the main constraint of the cutting-stock problem and many literatures only considered this constraint. The arranging strategy focuses on how to arrange the objects on the stock, and the arranging strategy will depend on the overlap consideration technique when arranging irregular objects. The arranging strategy considers how to obtain an arrangement pattern, and the search strategy is used to obtain another arrangement pattern to improve the arrangement result.

2.1 Combinational problem

The clustering-then-nesting strategy is useful for the combinational problem, and it is proposed by Cheng and Rao (1999). A sliding technique was proposed by Cheng and Rao (1997) to arrange all objects – the components – of a product first, i.e. clustering, then integrated these objects into a cluster, and nested it by CNA (Compact Neighborhood Algorithm). Finally, Cheng and Rao (2000) used a Genetic Algorithm

to adjust the position and orientation of the object to improve the result. The methods proposed by Cheng and Rao (2000) can be summarized as follows.

- Graph treatment: sliding technique.
- Arranging strategy: CNA.
- Search strategy: Genetic Algorithm.

2.2 Rectangular single-stock problem

Dowsland *et al.* (2002) used a “no-fit polygon” to obtain the closest position between two objects. The relative positions where one object contacts another object can be represented as a polygon called a “no-fit polygon”. They used a bottom-left strategy to arrange objects on the stock. The strategy of this literature can be summarized as follows:

- Graph treatment: no-fit polygon.
- Arranging strategy: bottom-left strategy.
- Search strategy: null.

In this method, the arrangement sequence governs the resulting arranging pattern. Thus, deciding the arrangement sequence is very important. Gomes and Oliveira (2002) changed the position of objects in the sequence to generate a new solution based on the original one. The strategy of Gomes and Oliveira (2002) can be summarized as follows:

- Graph treatment: no-fit polygon.
- Arranging strategy: bottom-left strategy.
- Search strategy: exchange.

Bennell and Dowsland (2001) formulated the cutting-stock problem as a Linear Programming model, and solved this model to obtain an arrangement pattern. The model led the objects move to the best position on the no-fit polygons. After an arrangement pattern was obtained, one object was moved to another position and the model was solved again. Bennell and Dowsland (2001) used Tabu Search was used to seek the global optimum solution in the solution space. The strategy used by Bennell and Dowsland (2001) can be summarized as follows:

- Graph treatment: no-fit polygon.
- Arranging strategy: Linear Programming.
- Search strategy: Tabu Search.

Gomes and Oliveira (2006) used bottom-left strategy to do the initial arrangement, and then swapped two objects randomly. The swapping will cause some overlaps, and these overlaps were relaxed by “separation step.” After relaxing the overlaps, the “compaction step” reduced the length of arrangement pattern. In the separation and compaction steps, the problem was formulated as different linear models to guide objects move to the good position on the no-fit polygons. If the

solution was improved after the compaction step, the solution was accepted. If the solution was not improved, Gomes and Oliveira (2006) the rule of Simulated Annealing Algorithm (SAA) was used to check acceptance or rejection of the solution. The SAA is a popular global optimization method. When using the SAA, various operators used to determine a new solution had to be designed for “searching”. The solution is updated to the new solution if the cost function is decreased. If the cost function of the new solution is larger than or equal to the original one, a random number σ will be generated as $0 \leqq \sigma \leqq 1$. If

$$\sigma \leq e^{\frac{-\Delta E}{T}} \tag{1}$$

where T is the temperature of the SAA, and ΔE is the increment of the cost function, the solution will be updated to a new one. The temperature here is not a real temperature. It is a parameter used to simulate a real annealing process, while the initial temperature, final temperature, and cooling rate have to be set when using the SAA. The strategy used by Gomes and Oliveira (2006) can be summarized as follows:

- Graph treatment: no-fit polygon.
- Arranging strategy: Linear Programming (separation and compaction steps).
- Search strategy: object swapping and SAA.

Poshyanonda and Dagli (2004) represented objects as binary matrices, and used an artificial neural network and the Genetic Algorithm to solve the cutting-stock

problem. The strategy used by Poshyanonda and Dagli (2004) can be summarized as follows:

- Graph treatment: binary matrix representation.
- Arranging strategy: artificial neural network.
- Search strategy: Genetic Algorithm.

Ratanapan *et al*. (2007) used an evolutionary algorithm to solve the cutting-stock problem. The evolutionary algorithm had a fitness function similar to the Genetic Algorithm, and has some operators, such as translation, rotation, touch point, relocate away, etc., designed by the authors to escape from the local optimum trap. The strategy used by Ratanapan *et al*. (2007) can be summarized as follows:

- Graph treatment: binary matrix representation.
- Arranging strategy: evolutionary algorithm.
- Search strategy: evolutionary algorithm.

Bouganis and Shanahan (2007) sorted the objects into a sequence according to their area and arranged an object on the position lead the smallest enclosing rectangle of arrangement pattern. This sorting method is called “object-based approach.” After arranging an object, the shape of the void region, i.e. the region not arranged with objects, was evaluated with the remainder objects for profile matching by computer vision method. If there is an object match the void region, the object will be arranged.

If not, the next object in the sequence will be arranged. This is called “scene-driven approach.” The method used in this literature can be summarized as follows:

- Graph treatment: binary matrix representation.
- Arranging strategy: smallest enclosing rectangle + object-based approach + scene-driven approach.
- Search strategy: null.

Egeblad *et al.* (2007) used bottom-left method to do the initial arrangement, and then reduced the total length of arrangement pattern. The reduction will cause some overlaps. The overlaps are relaxed by translating or rotating objects. The cost function in the relaxation step is the total overlap area of all objects. The cost function is express as a function of the horizontal positions of objects, and the overlap area is defined as the area between edges of two objects in the horizontal direction. After a new solution is obtained, the cost function is altered for escaping the local optimum trap. The strategy used by Egeblad *et al.* (2007) can be summarized as follows:

- Graph treatment: overlap area.
- Arranging strategy: overlap relaxation (translation and rotation).
- Search strategy: cost function alteration.

Burke *et al.* (2006) found the relationship between line to line, line to arc, arc to line and arc to arc. Thus they can use the real arc not approximation edges when

arranging objects to relax overlap. At beginning, an arranging sequence was decided, and objects are put into stock one by one. Every object was put on the bottom-left corner of the stock firstly, and used the relationship between lines and arcs to move the object to a position without overlap. After a solution was obtained, some objects were swapped in the arranging sequence to obtain new solution. Some solutions better than the current one were obtained and the best solution was updated as the new solution. The Tabu Search was used to guide the searching in the solution space. The Tabu Search recorded the solution history by recording the recent solutions with pre-decide length, for example 200 solutions, and the same solution as these 200 solutions can not be obtained again. The strategy used by Burke *et al.* (2006) can be summarized as follows:

- Graph treatment: relationship between lines and arcs
- Arranging strategy: bottom-left arranging + overlap relaxation
- Search strategy: swapping position in sequence + Tabu Search

2.3 Multi-stock problem

Babu and Babu (2001) coded the irregular stock and objects as an integer array. The cells that can arrange object on is coded as 0. The other cells are coded as an integer number that starts from 1 since the most right cell. Objects are coded in the opposite way and arranged near the bottom of the stock as close as possible. These

integer numbers are helpful for increasing the speed of finding the optimum arrangement. The method for searching a good arrangement sequence is the Genetic Algorithm coded with the stock number, the object number, and the object orientation. Zhang *et al.* (2005) used this method to design software for leather nesting. The strategy used in these two literatures can be summarized as follows:

- Object graph treatment: integer array representation.
- Multi-stock treatment: Genetic Algorithm.
- Arranging strategy: bottom-left strategy.
- Search strategy: Genetic Algorithm.

Wu *et al.* (2003) represented objects by binary matrices, and arranged objects on multiple stocks by bottom-left strategy. The stock was selected randomly when arranging every object. They swapped the object sequence and the stock sequence to obtain different solutions, and accepted a new solution by SAA rule shown in equation (1). The strategy used by Wu *et al.* (2003) can be summarized as follows.

- Object graph treatment: binary array representation.
- Multi-stock treatment: select randomly.
- Arranging strategy: bottom-left strategy.
- Search strategy: swap sequence + SAA.

2.4 Irregular stock problem

As introduced in section 2.3, the method proposed by Babu and Babu (2001) also considered the irregular stock problem. The strategy can be summarized as follows:

- Object graph treatment: integer array representation.
- Irregular stock treatment: integer array representation.
- Arranging strategy: bottom-left strategy.
- Search strategy: Genetic Algorithm.

Tay *et al.* (2002) arranged one object to contact the stock boundary with at least a vertex. The object can be slid and keep contact the boundary of the irregular stock to search the optimum position by the Genetic Algorithm. The genome of the Genetic Algorithm contained the object position along the stock boundary and the orientation of the object. The objects are arranged around the stock boundary sequentially. After arranging an object, the object profile is compounded with the profile of the irregular stock, and the compound is considered as a new irregular stock. The strategy can be summarized as follows:

- Object graph treatment: contact.
- Irregular stock treatment: slid and keep contact.
- Arranging strategy: Genetic Algorithm.
- Search strategy: null.

Huang *et al.* (2005) used an opposite strategy to arrange objects. They arranged objects “near to center,” and presented a rule called “best-matching algorithm” to decide the relative position of two objects. The first object is arranged on the center of the smallest enclose rectangle’s center of the irregular stock. After two objects are arranged, they are treated as a compound object and the others are arranged with this compound object sequentially. They used the 2-exchange procedure proposed by Gomes *et al.* (2002) to modify the object sequence to search the best one. The strategy used by Huang *et al.* (2005) can be summarized as follows:

- Object graph treatment: original shape + checked.
- Irregular stock treatment: original shape + checked.
- Arranging strategy: best-matching algorithm.
- Search strategy: 2-exchange procedure.

Crispin *et al.* (2005) found the no-fit polygon (NFP) between not only objects but an object and an irregular stock to consider the irregular profiles. And then used the Genetic Algorithm to obtain the arranging pattern. The strategy used by Crispin *et al.* (2005) can be summarized as follows:

- Object graph treatment: no-fit polygon.
- Irregular stock treatment: no-fit polygon.
- Arranging strategy: arranged on the crossing point of no-fit polygons.

- Search strategy: Genetic Algorithm.

Yuping *et al.* (2005) didn't consider the irregular characteristic of the stock by graph treatment but the penalty concept. They considered the cutting-stock problem as an unconstrained optimization problem. The design variables are positions and orientations of objects. The cost function had three parts. The major one is the area of the unplacement stock. The area of objects escaped from the stock, and the overlap area between objects are treated as penalty functions of the cost function. They generated random movements of objects and used the SAA to search the global optimum solution. The strategy used by Yuping *et al.* (2005) can be summarized as follows:

- Object graph treatment: penalty function with object overlap area.
- Irregular stock treatment: penalty function with escaped object area.
- Arranging strategy: random movement.
- Search strategy: random movement and SAA.

The methods use no-fit polygon as the graph treatment can be used to solve the cutting-stock problem when objects should be arranged in a special orientation, because the object orientation is fixed when finding the no-fit polygon.

When using binary matrices to represent objects, the object may be deformed if rotating the binary matrix instead of the real object for considering multiple

orientations. It costs much time to rotate real object because the object have to be re-coded.

The Genetic Algorithm is a popular and widely-use method for searching for the global optimum solution. However, it has many parameters that need to be decided, such as the crossover rate and the mutation rate. Also, each parameter setting will greatly affect the result, as shown by Poshyanonda and Dagli (2004). Every design variable is transferred to a binary code in Genetic Algorithm, and the resolution of the binary code will affect the result. This is another disadvantage of the Genetic Algorithm.

The SAA has the similar drawback to Genetic Algorithm. The parameter setting of the SAA is also important for obtaining a good solution as shown by Marques *et al.* (1991) and Leung *et al.* (2003). The parameter setting depends on experience very much and not easy to use.

CHAPTER 3 METHODS

The basic method for solving the cutting-stock problem in this study is formulating the problem as a constrained optimization problem and solving the constrained optimization problem by the Sequential Quadratic Programming (SQP) method. And the SQP method is used with different methods for different types of problem in this study. The methods used in this study are listed as follows.

1. Maximum depth method.
2. SQP method.
3. Compact Neighborhood Algorithm (CNA).
4. Parallism strategy.
5. Virtual object strategy.
6. Swap strategy.
7. Insert strategy.

The first method is used to consider the overlap between objects, and it overcomes the difficulty of the irregular profile of objects. The SQP method is used to arrange objects. The CNA and the parallism strategy are used for the combinational problem. The virtual object strategy is used for the irregular stock problem, multi-stock problem, and flaw stock problem. The swap strategy and the insert strategy are used to find the global optimum results. The methods used in this study

can be summarized as follows:

For combinational problem:

- Graph treatment: maximum depth method and SQP method.
- Arranging strategy: SQP method and CNA.
- Search strategy: parallism strategy

For other problems:

- Object graph treatment: maximum depth method and SQP method.
- Irregular stock treatment: virtual object.
- Multi-stock treatment: virtual object.
- Arranging strategy: SQP method.
- Search strategies: 1. swap strategy
  2. insert strategy.

The CNA and SQP method are published in the literatures. They are also introduced in this chapter because of the easy reading. As the same reason, only the maximum depth method and SQP method that are used in every solving process are introduced independently. Other methods are introduced in the solving process.

3.1 Maximum depth method

The objects of the cutting-stock problem cannot overlap one other. In traditional methods, the overlap area is calculated and the object positions are adjusted to reduce

the overlap area until the total overlap area equals to zero when considering overlap directly. This study uses the maximum "depth" of two objects as the overlap index when considering the overlap.

When considering overlap between two objects, the "depth" means the distance from the vertex on one object to a point on the edge of the other object, and the maximum depth is the largest distance. The detailed calculation process is shown in Fig. 3-1 while the process can be explained with the example in Fig. 3-2. A and B are two objects and every vertex is numbered in a counterclockwise order (a_1~a_8 and b_1~b_8 , respectively). O_A and O_B are the reference points of object A and object B respectively. It is used to represent the position of the object. The centres of gravity (COG) is used as the reference point in this study. As shown in the process (Fig. 3-1), the first step of finding the maximum depth is transferring the original coordinate to the coordinate system where the y-direction is parallel to the $\overrightarrow{O_AO_B}$ vector, which will be helpful for calculating depths. The next step initialises the maximum depth, and its value is determined by subtracting the y-value of a_1 from the y-value of a_1', where a_1' is the point that a_1 projects onto object B in the $\overrightarrow{O_AO_B}$ direction. The depth will be positive if the vertex is inside the other object, such as the depth of a_1. Similarly, if the vertex is outside the other object, the depth will be negative. Once the depth of a_1 is known, the depth of a_2 will be calculated, and is less than the depth of a_1

as shown in the figure. Thus, the maximum depth will not be updated. Only the depths of the object vertices have to be calculated when finding the maximum depth, and it is not necessary to calculate the depths of the points on the edge. It is obvious that the maximum depth will coincide with a vertex, because all edges are linear. If depth of a point is searched along an edge, it will be increased or decreased monotonously until the movement meets a corner. Thus, the maximum depth will coincide with a vertex. The next point is a_3 and its depth is negative because the y-value of a_3 is larger than a_3'. Other vertices on object A will also be considered one by one. Similarly, the depths of the vertices on object B are determined by subtracting the y-value of the projection point from the y-value of the vertex, and calculated one by one.

By this way, the depth will be negative if the vertex is outside of the other object, and it is not necessary to check the vertex is inside the other object or not. The overlap index will be negative if there is a gap between two objects. A negative overlap index will be helpful for the active-set strategy of SQP method. The maximum depth will be negative but the overlap area is never negative. Therefore, the maximum depth is more suitable for the SQP method with the active-set strategy, which will be described in the next section.

3.2 Sequential Quadratic Programming method

The SQP method is a numerical method for solving optimization problems. The

procedure of a numerical method is an iterative process of finding a "search direction" and a "step size".

To solve the optimization problem by the SQP method, the Karush-Kuhn-Tucker (KKT) conditions that is introduced by Arora (2004) of the Lagrange function are used. The Lagrange function is defined as follows:

$$L(\mathbf{d},\boldsymbol{\mu}) = f(\mathbf{d}) + \boldsymbol{\mu}^T \mathbf{g} \tag{2}$$

where $\mathbf{d}$ is the vector form of design variables; $\boldsymbol{\mu}$ is the vector form of the Lagrange multipliers; $\mathbf{g}$ is the vector form of constraints. The numerical solving process of the KKT conditions is an iterative process of calculating the new solution $\mathbf{d}^{(k+1)}$,

$$\mathbf{d}^{(k+1)} = \mathbf{d}^{(k)} + \Delta\mathbf{d}^{(k)} \tag{3}$$

where k is the iteration number, and $\Delta\mathbf{d}^{(k)}$ is the change in design variables. It is also the search direction of the SQP method. The SQP method defines a QP sub-problem to calculate the search direction. The flowchart of the SQP method is shown in Fig. 3-3, and the process is described as follows. The description are summarized from the paper written by Arora (1984):

1. Select an initial solution, and set other parameters that will be described later.
2. Define the QP sub-problem. The QP sub-problem is defined as follows:

   minimize $\nabla f^T \Delta\mathbf{d} + 0.5\Delta\mathbf{d}^T \mathbf{H} \Delta\mathbf{d}$ (4)

   subject to $\mathbf{A}^T \Delta\mathbf{d} \le \mathbf{b}$ (5)

where **H** is the approximate Hessian matrix of the Lagrange function, and

$$A_{j_i} = \frac{\partial g_{ai}}{\partial d_j} \tag{6}$$

$$b_i = -g_{ai}(\mathbf{d}) . \tag{7}$$

When defining the QP sub-problem, only the constraints that are larger than or equal to zero need to be considered. The g_{ai} is the *i*-th element of $\mathbf{g_a}$ which is the vector form of these constraints. This is the so-called active-set strategy.

3. After the QP sub-problem is defined, the problem is normalized before solving. For normalizing the QP sub-problem, the **H** is decomposed as

$$\mathbf{H} = \mathbf{U}^T\mathbf{U} \tag{8}$$

where **U** is an upper triangular matrix. A new variable **s** is defined as

$$\mathbf{s} = \mathbf{U}\Delta\mathbf{d} \tag{9}$$

and the problem can be normalized as:

minimize $\quad \mathbf{\Lambda_0}^T\mathbf{s} + 0.5\mathbf{s}^T\mathbf{s}$ (10)

Subject to $\quad \mathbf{Ms} \le \mathbf{\Delta}$ (11)

where $\quad \mathbf{\Lambda_0} = \dfrac{\mathbf{U}^{-1^T}\nabla f}{c_0}$ (12)

$$c_0 = \left\| \mathbf{U}^{-1^T}\nabla f \right\| \tag{13}$$

$$\Delta_i = \frac{-g_{ai}}{c_i} \tag{14}$$

$$c_i = \left\| \mathbf{U}^{-1^T}\nabla \mathbf{g_{ai}} \right\| \tag{15}$$

$$\mathbf{M} = \frac{\mathbf{U}^{-1^T}\mathbf{A}}{c_0} \tag{16}$$

4. Before solving the normalized QP sub-problem, the maximum value of constraints is compared with a constant ε_c. The maximum value of constraints is defined as

$$F(\mathbf{d}) = \max\{0, g'_1(\mathbf{d}), ..., g'_{p'}(\mathbf{d})\} \tag{17}$$

If the maximum value of constraints is less than ε_c, i.e., the violation is not serious, the solution will focus on reducing the cost function. The solution is:

$$\mathbf{v} = \mathbf{v}_1 + \mathbf{v}_2 \tag{18}$$

$$\mathbf{s} = -\mathbf{s}_1 + \mathbf{s}_2 \tag{19}$$

where

$$\mathbf{v}_1 = -\mathbf{B}^{-1}\mathbf{M}^T\boldsymbol{\Lambda}_0 \tag{20}$$

$$\mathbf{v}_2 = -\mathbf{B}^{-1}\boldsymbol{\Lambda} \tag{21}$$

$$\mathbf{B} = \mathbf{M}^T\mathbf{M} \tag{22}$$

$$\mathbf{s}_1 = \boldsymbol{\Lambda}_0 + \mathbf{M}\mathbf{v}_1 \tag{23}$$

$$\mathbf{s}_2 = -\mathbf{M}\mathbf{v}_2 \tag{24}$$

If not, the solution focuses on correcting the constraints. The solution is:

$$\mathbf{v} = \mathbf{v}_2 \tag{25}$$

$$\mathbf{s} = \mathbf{s}_2 \tag{26}$$

5. The solution of the original QP sub-problem can be obtained as:

$$\mu_i = \frac{c_0 v_{1i} + v_{2i}}{c_i} \tag{27}$$

$$\boldsymbol{\Delta}\mathbf{d} = \mathbf{U}^{-1}\mathbf{s} \tag{28}$$

6. Check the stop condition. If $\left\|\mathbf{d}^{(k)}\right\| \le \varepsilon$ or $k>k_{\max}$, stop the process and the current solution is the final solution. If not, continue the process. ε is a preset small number close to zero, and $k_{\max}$ is the maximum iteration number set in the initialization step.

7. Calculate the step size. The step size α_k is set as

$$\alpha_k = 0.5^q, \quad q = 0,1,2,... \tag{29}$$

The minimum q that makes $\Phi(\mathbf{d} + 0.5^q \Delta\mathbf{d}) \le \Phi(\mathbf{d})$ (30)

is used to define α_k

where $\Phi(\mathbf{d}) = f(\mathbf{d}) + R_k F(\mathbf{d})$ (31)

$$R_k = 0.5(R_{k-1} + \sum_{i=1}^{p'} \mu_i) \tag{32}$$

8. Update the approximate Hessian matrix. The approximate Hessian matrix is updated by the BFGS strategy that is introduced by Arora (2004.) The BFGS strategy is described as follows:

Define three variables first.

$$\mathbf{z}^{(k)} = \alpha_k \mathbf{H}^{(k)} \Delta\mathbf{d}^{(k)} \tag{33}$$

$$\mathbf{u}^{(k)} = \nabla L(\mathbf{d}^{(k+1)}, \boldsymbol{\mu}^{(k)}) - \nabla L(\mathbf{d}^{(k)}, \boldsymbol{\mu}^{(k)}) \tag{34}$$

and $\mathbf{w}^{(k)} = \psi \mathbf{u}^{(k)} + (1-\psi)\mathbf{z}^{(k)}$ (35)

where $$\psi = \begin{cases} 1, & \text{if } \Delta\mathbf{d}^{(k)^T}\mathbf{u}^{(k)} \ge \Delta\mathbf{d}^{(k)^T}\mathbf{z}^{(k)} \\ 0.8, & \text{otherwise} \end{cases} \tag{36}$$

Then, the approximate Hessian matrix is updated as

$$\mathbf{H}^{(k+1)} = \mathbf{H}^{(k)} + \mathbf{P}^{(k)} - \mathbf{E}^{(k)} \tag{37}$$

where

$$\mathbf{P}^{(k)} = \frac{\mathbf{w}^{(k)}\mathbf{w}^{(k)^T}}{\alpha_k \Delta\mathbf{d}^{(k)^T}\mathbf{w}^{(k)}} \tag{38}$$

$$\mathbf{E}^{(k)} = \frac{\mathbf{z}^{(k)}\mathbf{z}^{(k)^T}}{\alpha_k \Delta\mathbf{d}^{(k)^T}\mathbf{z}^{(k)}} \tag{39}$$

9. Update the solution as

$$\mathbf{d}^{(k+1)} = \mathbf{d}^{(k)} + \alpha_k \Delta\mathbf{d}^{(k)} \tag{40}$$

and continue to go to step 2 to define the QP sub-problem.

All equations are shown above, and the derivation can be found in the work by Arora (2004) and Liao (1990). There are several programs, such as MOST that is designed by Tseng (1989) and IDESIGN that is designed by Arora (1988,) that use the SQP method to solve the constrained optimization problem.

3.3 The solving process for the combinational problem

The solving process for the combinational problem can be divided into three steps in this study. The first step is to arrange the different objects as a cluster, and then to generate the nesting pattern according to the cluster. Finally, the third step is to adjust the orientation of the nesting pattern that generated in the second step to maximize the stock utility rate.

3.3.1 Clustering

The cutting-stock problem is formulated as a standard form of the constrained optimization problem as follows:

cost function: $$\text{minimize} \quad f = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \left\| \overrightarrow{O_i O_j} \right\| \tag{41}$$

design variables: $$x_i, y_i, \theta_i$$

$$i=1 \sim N \tag{42}$$

constraints: $$g1_i = D_{Mijk}(x_j, y_j, \theta_j, x_k, y_k, \theta_k) \le 0$$

$$i = 1 \sim \frac{N(N-1)}{2}$$

$$j=1 \sim (N-1)$$

$$k=(j+1) \sim N \tag{43}$$

where N is the number of objects; $\left\| \overrightarrow{O_i O_j} \right\|$ is the norm of vector $\overrightarrow{O_i O_j}$, i.e., the distance between O_i and O_j; x_i is the x-coordinate value of the reference point of object i; y_i is the y-coordinate value of the reference point of object i; θ_i is the orientation of object i; D_{Mjk} is the maximum depth between object j and object k. The cost function is to minimize the summation of distances between any two objects, which means that the objects have to be arranged as close as possible. This cost function (the distance summation) is more sensitive to the design variables than the number of objects or the necessary stock area. Therefore, it is more suitable for the SQP method. The constraints are that the maximum depths of any two objects cannot

be larger than zero, i.e., one object can only be far away or just contact the adjacent object.

As shown in equation (43), there are many constraints when the number of objects is large, and reducing the number of constraints is important to decrease the calculation effort. Because of the nature of the cutting-stock problem, the constraints may not be reduced in physical ways, but they can be reduced using mathematical methods. If the solution satisfies the constraint, it will not be necessary to consider whether or not the constraint still exists. Therefore, the active-set strategy of the SQP method is used to reduce the considered constraints in this study.

A constraint will never be inactive when using the overlap area as the constraint value because the overlap area is never less than zero. Thus, the number of constraints cannot be reduced. However, when using the maximum depth to consider overlap, the maximum depth will be negative if there is a gap between two objects, and therefore the constraint becomes inactive. These constraints will be ignored and the computation effort will be decreased.

After formulating the clustering process as a constrained optimization problem, the positions and orientations of objects can be obtained by using SQP method to solve the problem.

3.3.2 Nesting

After the first step, the objects are arranged at the positions (x_i, y_i) with the orientations (θ_i), and are treated as a cluster in this step. This step uses the CNA method which proposed by Cheng and Rao (1999) to generate the nesting pattern. The detailed process is introduced with pentagons as an example as follows.

At first, two objects are in contact to each other at a point as shown in Fig. 3-4(a), and are called "stator" and "mover". The object S in Fig. 3-4(a) is the stator and the object M is the mover. The bottom-left vertex of the object is used to represent the object position in CNA. Then M slides on S with a fixed orientation in a counterclockwise direction. At beginning, m_4 contacts with s_1, and M will moves along $\overrightarrow{s_1s_2}$ until the contact vertex m_4 meet a corner or M contacts S on another point as shown in Fig. 3-4(b). The point m_4 meets a corner when it contacts with s_2, and then M will moves along $\overrightarrow{m_5m_4}$ until the contact vertex s_2 meet a corner or M contacts S on another point as shown in Fig. 3-4(c). This self-sliding process is complete when M moves to the initial position, and the path of the bottom-left vertex of M is recorded as a no-fit polygon as shown in Fig. 3-4(d). This process is called "self-sliding" because the two objects are identical and sliding relative to each other in the process.

After finding the no-fit polygon of self-sliding, the mover is removed and the

stator is called object P0 as shown in Fig. 3-5(a). The bottom-left vertex of another object (object P1) is put at an arbitrary position of the self-sliding no-fit polygon of P0. These two objects have their self-sliding no-fit polygons, and the right interaction point of these two no-fit polygons is the position of object P2. The vector from the bottom-left vertex of object P0 to the object P1 is called the "first nesting vector", and the vector from the bottom-left vertex of object P0 to the object P2 is the second nesting vector as shown in Fig. 3-5(b). The third nesting vector is obtained by subtracting the first nesting from the second nesting vector. These three nesting vectors and their negative vectors will form a hexagon as shown in Fig. 3-5(c), which is called a "nesting crystal". Then, moving object P1 on the self-sliding no-fit polygon of P0 will result in different nesting vectors and different nesting crystals. The optimum nesting vectors are those vectors that cause the minimum nesting crystal area.

3.3.3 Improvement

Because the optimum nesting vectors in Fig. 3-5(b) might not be parallel to the sheet stock edges, there will be four corners that are not occupied by the nesting pattern after the second step. Rotating the nesting vector parallel to any axis will reduce this to two regions. Aligning three nesting vectors parallel to the X- and Y-axis respectively will result in six cases to improve the nesting pattern. The best case is

selected as the result in this study.

3.4 The solving process for the multi-polygon problem

The multi-polygon problem includes five kinds of problems. They are the single-stock problem, multi-stock problem, rectangular stock problem, irregular stock problem, and flaw stock problem. The solving process for the multi-polygon problem is similar to the clustering step in the combinational problem. It formulates the multi-polygon problem as a constrained optimization problem and solves it by SQP method. One difference is that the virtual object is used in the model. The other difference is that a global strategy will be used after solving by SQP method. The global strategy is used to improve the solution, and there are two global strategies proposed in this study.

3.4.1 Virtual object strategy

When using the virtual object in the irregular stock problem, the smallest surrounding rectangle of the stock will be obtained first as shown in Fig. 3-6. Then the space between the profile of the stock and the surrounding rectangle is divided into several no-overlapping pieces as shown in Fig. 3-6. The objects arranged on the stock cannot have any part outside the boundary of the stock. This situation is equivalent to that no object can overlap with the pieces described above. Arranging objects on the

irregular stock is equivalent to arranging objects on the surrounding rectangle where the pieces are already arranged on. The pieces are called the “virtual” objects because they are not the real objects that are wanted to be arranged on the stock.

When using the virtual object in the multi-stock problem, the stocks are firstly arranged sequentially as shown in Fig. 3-7. The image of the arrangement of these stocks is similar to a large rectangular stock with some line segments. The objects that will be arranged on these stocks cannot be portioned into many parts in different stocks. Thus, the objects cannot overlap with the line segments on the large rectangular stock. Arranging objects on the stocks is equivalent to arranging objects on the large rectangular stock which has some line segments on it. The line segments are the virtual objects in the multi-stock problem.

When using the virtual object in the flaw stock problem, the flaw on the stock is marked first. And then, a virtual object with the same profile of the flaw is fixed on the position of the flaw before arranging objects. The object will be avoided to arrange on the flaw because they can not overlap the virtual object.

The cutting-stock problem with irregular stocks, multi-stock, or flaw stock is simplified as arranging objects on a rectangular single-stock where some virtual objects are already arranged on, and the objects cannot overlap with the virtual objects.

3.4.2 Problem formulation

The cutting-stock problem is formulated as a standard form of the constrained optimization problem as follows:

cost function: minimize $f = \sqrt{\sum_{i=1}^{N} x_{ui}^2}$ (44)

design variables: x_i, y_i, θ_i; i=1~N (45)

constraints: $g1_i = D_{Mijk}(x_j, y_j, \theta_j, x_k, y_k, \theta_k) \le 0$;

where $i = 1 \sim \frac{N(N-1)}{2}$, j=1~(N-1), k=(j+1)~N (46)

$g2_i = -x_{li} \le 0$; i=1~N (47)

$g3_i = -y_{li} \le 0$; i=1~N (48)

$g4_i = y_{ui} \le y_{width}$; i=1~N (49)

$g5_i = D_{Mijk} \le 0$; i=1~N×N_v, j=1~N, k=1~N_v (50)

where N is the number of objects; x_i is the x-coordinate value of the reference point of object i; y_i is the y-coordinate value of the reference point of object i; θ_i is the orientation of object i; x_{li} is the lower boundary of object i in the x-direction; x_{ui} is the upper boundary of object i in the x-direction; y_{li} is the lower boundary of object i in the y-direction; y_{ui} is the upper boundary of object i in the y-direction; y_{width} is the width of the stock; D_{Mjk} is the maximum depth between object j and object k; N_v is the number of virtual objects.

The cost function will cause objects be arranged near the lower boundary of the

stock as close as possible. The design variables will denote the positions and the orientations of objects. A reference point is used to denote the position of an object as described in section 3.1. The maximum depth D_{Mjk} between object j and object k is the maximum overlap length in the direction from the reference point of object j to the reference point of object k. The D_{Mjk} will be negative if two objects have no overlap but a gap between them. The constraints $g1_i$ can be shown as a vector form **g1**, and it considers the overlap of objects. The vector form **g2**, **g3**, and **g4** are similar to **g1**, and they constrain the objects to avoid being arranged outside the boundaries of the surrounding rectangular stock in the irregular stock problem and the large rectangular stock in the multi-stock problem. The equations of **g5** are the same definition in **g1** and it considers the overlap between objects and virtual objects.

The gradients are necessary in the local search strategy of this study, but the gradients of **g1** and **g5** do not have explicit forms. They will be calculated by the finite difference method in this study. The equation of the finite difference method used in this study is shown as follows.

$$\frac{\partial gi_j}{\partial d_k} = \frac{gi_j(\mathbf{d}+\Delta\mathbf{d}) - gi_j(\mathbf{d})}{\Delta d_k};\ k=1\sim N\times 3 \tag{51}$$

where **d** is the vector form of design variables and d_k is an element of **d**; and

$$\mathbf{d}+\Delta\mathbf{d} = \{d_1,...,d_k+\Delta d_k,...,d_{N\times 3}\}^T \tag{52}$$

Because of the characteristic of this formulation, much calculation of the gradient in

equation (51) is not necessary. For example, when modifying the design variables of object No. 3, it will never affect the overlap between object No.1 and object No. 2. Therefore

$$\frac{\partial D_{M12}}{\partial x_3} = \frac{\partial D_{M12}}{\partial y_3} = \frac{\partial D_{M12}}{\partial \theta_3} = 0 \tag{53}$$

and calculating them by equation (51) is not necessary.

Therefore the gradients of the constraints are shown as follows.

$$\frac{\partial g1_i}{\partial d_j} = \begin{cases} \dfrac{\partial D_{Mikl}}{\partial d_j}; \text{ when } d_j = x_k, y_k, \theta_k, x_l, y_l, \text{ or } \theta_l \\ 0; \quad \text{otherwise} \end{cases} \tag{54}$$

$$\frac{\partial g2_i}{\partial d_j} = \begin{cases} -\dfrac{\partial x_{li}}{\partial d_j}; \text{ when } d_j = \theta_i \\ -1; \quad \text{when } d_j = x_i \\ 0; \quad \text{otherwise} \end{cases} \tag{55}$$

$$\frac{\partial g3_i}{\partial d_j} = \begin{cases} -\dfrac{\partial y_{li}}{\partial d_j}; \text{ when } d_j = \theta_i \\ -1; \quad \text{when } d_j = y_i \\ 0; \quad \text{otherwise} \end{cases} \tag{56}$$

$$\frac{\partial g4_i}{\partial d_j} = \begin{cases} \dfrac{\partial y_{ui}}{\partial d_j}; \text{ when } d_j = \theta_i \\ 1; \quad \text{when } d_j = y_i \\ 0; \quad \text{otherwise} \end{cases} \tag{57}$$

$$\frac{\partial g5_i}{\partial d_j}=\begin{cases}\dfrac{\partial D_{Mikl}}{\partial d_j};\ \text{when } d_j=x_k, y_k, \text{ or } \theta_k\\ 0; \qquad \text{otherwise}\end{cases} \tag{58}$$

For simplifying the calculation of overlap, the real maximum depth will not be calculated if the smallest surrounding rectangles of object j and k have no overlaps. At this time, the minus value of the distance between two objects' reference point is used instead of the maximum depth. It is used for the **g1** and **g5**, and they are modified as follows.

$$g1_i'=\begin{cases}-\sqrt{(x_j-x_k)^2+(y_j-y_k)^2}\le 0,\\ \qquad \text{when } x_{lj}>x_{uk} \text{ or } x_{uj}<x_{lk} \text{ or } y_{lj}>y_{uk} \text{ or } y_{uj}<y_{lk};\\ D_{Mijk}(x_j,y_j,\theta_j,x_k,y_k,\theta_k)\le 0, \text{ otherwise}\end{cases}$$

$$\text{where } i=1\sim\frac{N(N-1)}{2}, j=1\sim(N\text{-}1), k=(j+1)\sim N \tag{59}$$

$$g5_i'=\begin{cases}-\sqrt{(x_j-x_k)^2+(y_j-y_k)^2}\le 0,\\ \qquad \text{when } x_{lj}>x_{uk} \text{ or } x_{uj}<x_{lk} \text{ or } y_{lj}>y_{uk} \text{ or } y_{uj}<y_{lk};\\ D_{Mijk}(x_j,y_j,\theta_j,x_k,y_k,\theta_k)\le 0, \text{ otherwise}\end{cases}$$

$$\text{where } i=1\sim N\times N_v, j=1\sim N, k=1\sim N_v \tag{60}$$

The equations of gradient are modified as follows.

$$\frac{\partial g1_i'}{\partial d_l}=\begin{cases}\dfrac{g1_i'(\mathbf{d}+\Delta\mathbf{d})-g1_i'(\mathbf{d})}{\Delta d_l};\ \text{when } d_l=x_j, y_j, \theta_j, x_k, y_k, \text{ or } \theta_k\\ 0; \quad \text{otherwise}\end{cases} \tag{61}$$

$$\frac{\partial g5_i'}{\partial d_l}=\begin{cases}\dfrac{g5_i'(\mathbf{d}+\Delta\mathbf{d})-g5_i'(\mathbf{d})}{\Delta d_j};\ \text{when } d_l=x_j, y_j, \text{ or } \theta_j\\ 0; \quad \text{otherwise}\end{cases} \tag{62}$$

where j and k in equation (61) and (62) are associated with what they are in equation

(59) and (60).

3.4.3 SQP method with the swap strategy

The swap strategy is one of the global strategies in this study. It includes "escaping the local optimum trap" and "sometimes accepting a bad solution". Two objects will be swapped after finding a local optimum solution, and objects will be re-arranged by the SQP method. This will help to search for a solution in another region and escape the local optimum trap. If there is no solution better than the original one after several swaps, the best one in these swaps will be updated as the new solution. This is similar to the Simulated Annealing method that accepts a bad solution according to equation (1) described above.

The whole approach is shown in Fig. 3-8, and includes the following steps:

1. Arrange all objects on the sheet stock randomly, and decide the maximum iteration number. The random arrangement is the initial solution. The index "*IterNo*" indicates the iteration number now, and it is set as 0 at the beginning.
2. Use the SQP method to arrange objects, and the solution "**D**" and the cost function value "*F*" are obtained. And then, initialize the best solution "$\mathbf{D}_{best}$" and the best cost function value "F_{best}" as **D** and *F*.
3. Update the "*IterNo*" and initialize the "*SwapNo*". Set the maximum swap number as the iteration number.

4. Update the "*SwapNo*" first, and swap the position of two objects of solution **D**. The two objects are selected randomly.
5. There will be some overlap after swap two objects. Thus the SQP method is used to re-arrange all objects, and the solution of the swap sub-process "$\mathbf{D}_{\mathbf{sub}}$" and its cost function value "F_{sub}" are obtained.
6. If it is the first swap or the result of swap is better than the best solution in the swap sub-process, the best solution of the swap sub-problem "$\mathbf{D}_{\mathbf{subbest}}$" and its cost function value "$F_{subbest}$" are updated.
7. If "$F_{subbest}$" is larger than "F_{best}", i.e., the best solution of the swap sub-process is worse than the best one of total process, and it will be better to try another swap, or "*SwapNo*" is less than "*SwapMax*", i.e., another swap is allowable, go to step 4 to do another swap.
8. If another swap is not necessary or not allowable in the swap sub-process, check if "$F_{subbest}$" is better than "F_{best}" or not. If a solution better than the best one of the total process is obtained in the swap sub-process, update the best solution and its cost function.
9. If "*IterNo*" is less than "*IterMax*", update the swap base "**D**" as the best solution of the swap sub-process and go to step 3 to continue the process. If not, the best solution is the final solution.

The solving process of the SQP method with the swap strategy can be written as an algorithm form as follows:

Set *IterMax*
Set *IterNo*=0
Set initial solution of SQP method: **d** randomly
    where $\mathbf{d}=[x_1,y_1,\theta_1,x_2,y_2,\theta_2,...,x_N,y_N,\theta_N]=[d_1,d_2,...d_{Nx3}]^T$
Solve min f(**d**) subject to g(**d**)$\leqq$0 by SQP method
    Then obtain solution **D**, F=f(**D**)

Set $\mathbf{D_{best}}=\mathbf{D}$
Set $F_{best}=F$

For *IterNo*=1,2,...,*IterMax*
    Select a and b randomly where $1\leqq a\leqq N$, $1\leqq b\leqq N$
    Set initial solution of SQP method:
        $\mathbf{d}=[D_1,D_2,...,D_{ax3-4},D_{ax3-3},D_{bx3-2},D_{bx3-1},D_{ax3},...,D_{bx3-4},D_{bx3-3},D_{ax3-2},D_{ax3-1},D_{bx3},...,D_{Nx3}]^T$
    Solve min f(**d**) subject to g(**d**)$\leqq$0 by SQP method
        Then obtain solution $\mathbf{D_{sub}}$, F_{sub}=f($\mathbf{D_{sub}}$)

    Set $\mathbf{D_{subbest}}=\mathbf{D_{sub}}$
    Set $F_{subbest}=F_{sub}$

    Set SwapMax
    Set SwapNo=1

    While (*SwapNo*<*SwapMax* and $F_{subbest}>F_{best}$)
        Select a and b randomly where $1\leqq a\leqq N$, $1\leqq b\leqq N$
        Set initial solution of SQP method:
            $\mathbf{d}=[D_1,D_2,...,D_{ax3-4},D_{ax3-3},D_{bx3-2},D_{bx3-1},D_{ax3},...,D_{bx3-4},D_{bx3-3},D_{ax3-2},D_{ax3-1},D_{bx3},...,D_{Nx3}]^T$
        Solve min f(**d**) subject to g(**d**)$\leqq$0 by SQP method
            Then obtain solution $\mathbf{D_{sub}}$, F_{sub}=f($\mathbf{D_{sub}}$)

        If $F_{sub}<F_{subbest}$
            Set $\mathbf{D_{subbest}}=\mathbf{D_{sub}}$

```
            Set F_subbest=F_sub
        End If

        Set SwapNo=SwapNo+1
    End While

    Set D=D_subbest

    If F_subbest<F_best
        Set D_best=D_subbest
        Set F_best=F_subbest
    End If
End For

Obtain D_best as the solution
```

The maximum swapping number is set as the iteration number, because the bad solution may be accepted easily in the beginning of the solving process. The acceptance of bad solution will become more and more difficult in the solving process. This characteristic is similar to the concept of Simulated Annealing Algorithm for global searching. As the maximum swapping number is increased during every iteration in the process, the number of bad solution acceptances is decreased. It is similar to the "cooling down" in the Simulated Annealing Algorithm, but no additional parameter has to be set, such as the temperature and the cooling rate of the Simulated Annealing Algorithm and the mutation rate of the Genetic Algorithm. It is friendly and easy to use.

3.4.4 SQP method with the insert strategy

As shown in the equation (1), the objects are desired to be arranged near the lower boundary in the x-direction as close as possible. After arranging by the local search strategy, the object with the maximum upper boundary in the x-direction is inserted to the x-lower boundary of the stock with the same height. This will cause the solution go into another local region but also produce some overlaps.

The flowchart of the whole solving process is shown in Fig. 3-9. The steps are described below.

1. Arrange objects on the stock randomly and it is the initial solution of the whole process. Choose the maximum iteration number "*IterMax*" and set the iteration number index "*IterNo*" as 0.
2. Improve the initial solution with SQP method.
3. Initialize the best solution "$\mathbf{D}_{\mathbf{best}}$" as the solution of the SQP method.
4. Insert the object with maximum x-upper bound into the x-lower boundary of the stock.
5. Improve the overlap by using SQP method.
6. Update $\mathbf{D}_{\mathbf{best}}$. If the necessary length, i.e. the maximum x-upper bound of all objects, of the stock is reduced, set the new solution as the best solution.

If the iteration number "*IterNo*" is less than the maximum iteration number, add

IterNo by 1 and go to step 4. Otherwise stop the process and the $\mathbf{D}_{\mathbf{best}}$ is the best solution.

The solving process of the SQP method with the insert strategy can be written as an algorithm form as follows:

Set *IterMax*
Set *IterNo*=0
Set initial solution of SQP method: **d** randomly
    where $\mathbf{d}=[x_1,y_1,\theta_1,x_2,y_2,\theta_2,...,x_N,y_N,\theta_N]=[d_1,d_2,...d_{Nx3}]^T$
Solve min f(**d**) subject to g(**d**)$\leqq$0 by SQP method
    Then obtain solution **D**, F=f(**D**)

Set $\mathbf{D}_{\mathbf{best}}=\mathbf{D}$
Set $F_{best}=F$

For *IterNo*=1,2,...,*IterMax*
    Set $x_{ui}=\text{Max}\{x_{u1},x_{u2},...,x_{uN}\}$ and obtain i
    Set $\mathbf{d}=[D_1,D_2,...,D_{ax3-3},0,D_{ax3-1},...,D_{Nx3}]^T$
    Solve min f(**d**) subject to g(**d**)$\leqq$0 by SQP method
    Then obtain solution **D**, F=f(**D**)

    If $F<F_{best}$
        Set $\mathbf{D}_{\mathbf{best}}=\mathbf{D}$
        Set $F_{best}=F$
    End If
End For

Obtain $\mathbf{D}_{\mathbf{best}}$ as the solution

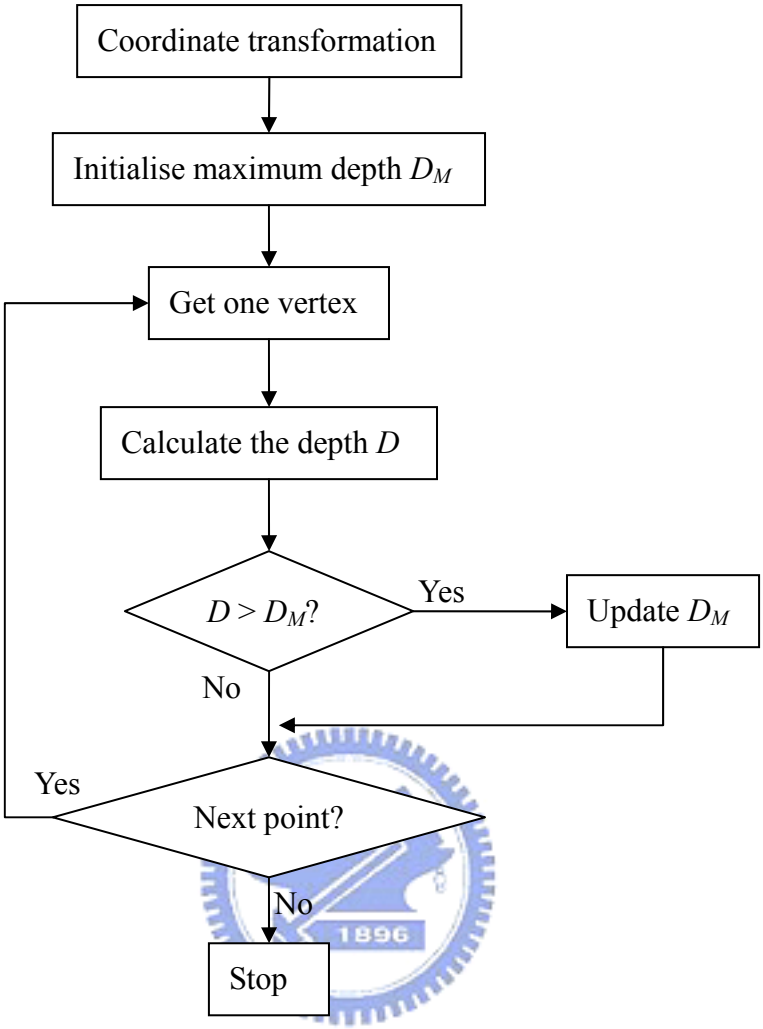


Fig. 3-1 Flowchart for finding the maximum depth

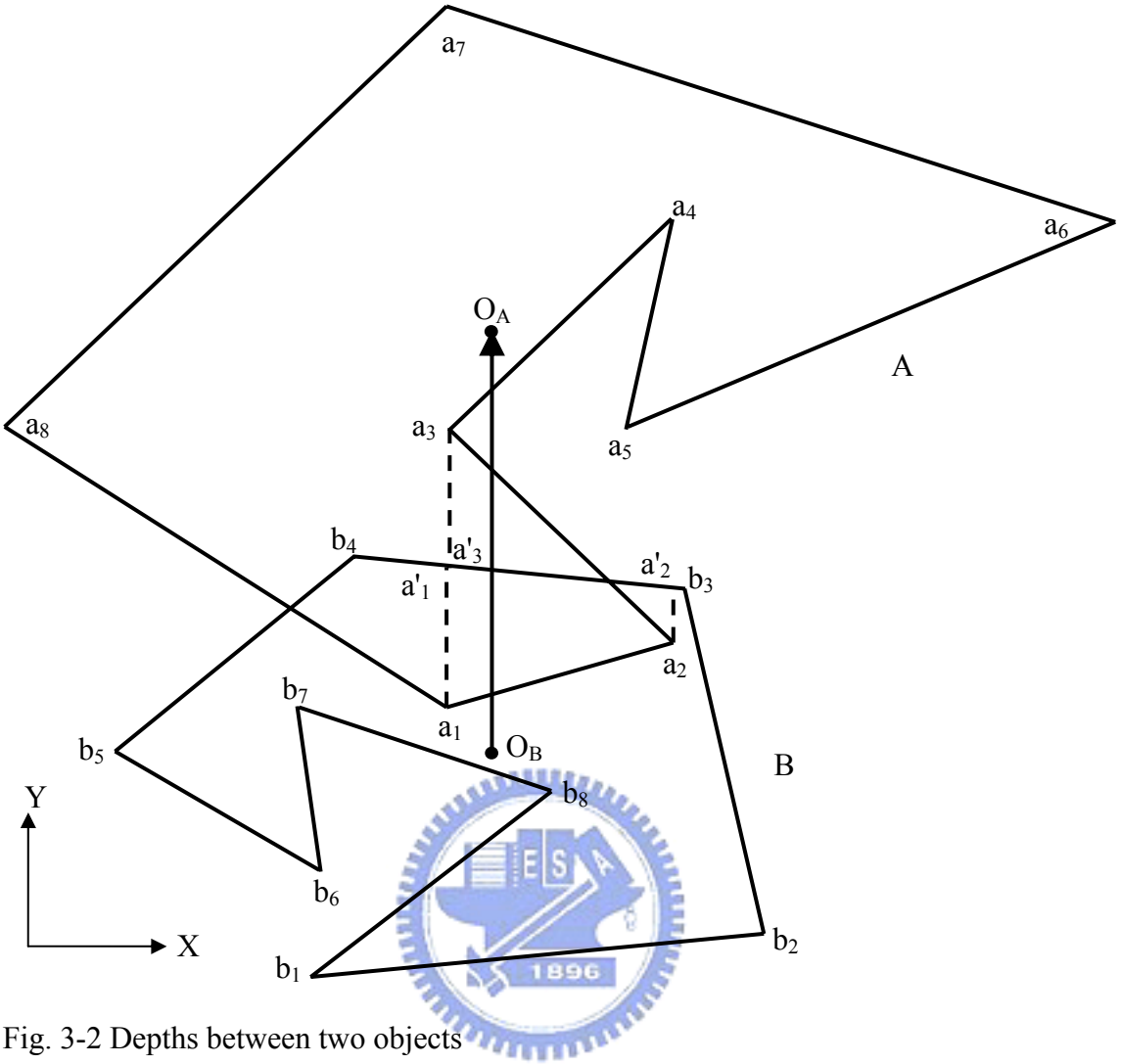


Fig. 3-2 Depths between two objects

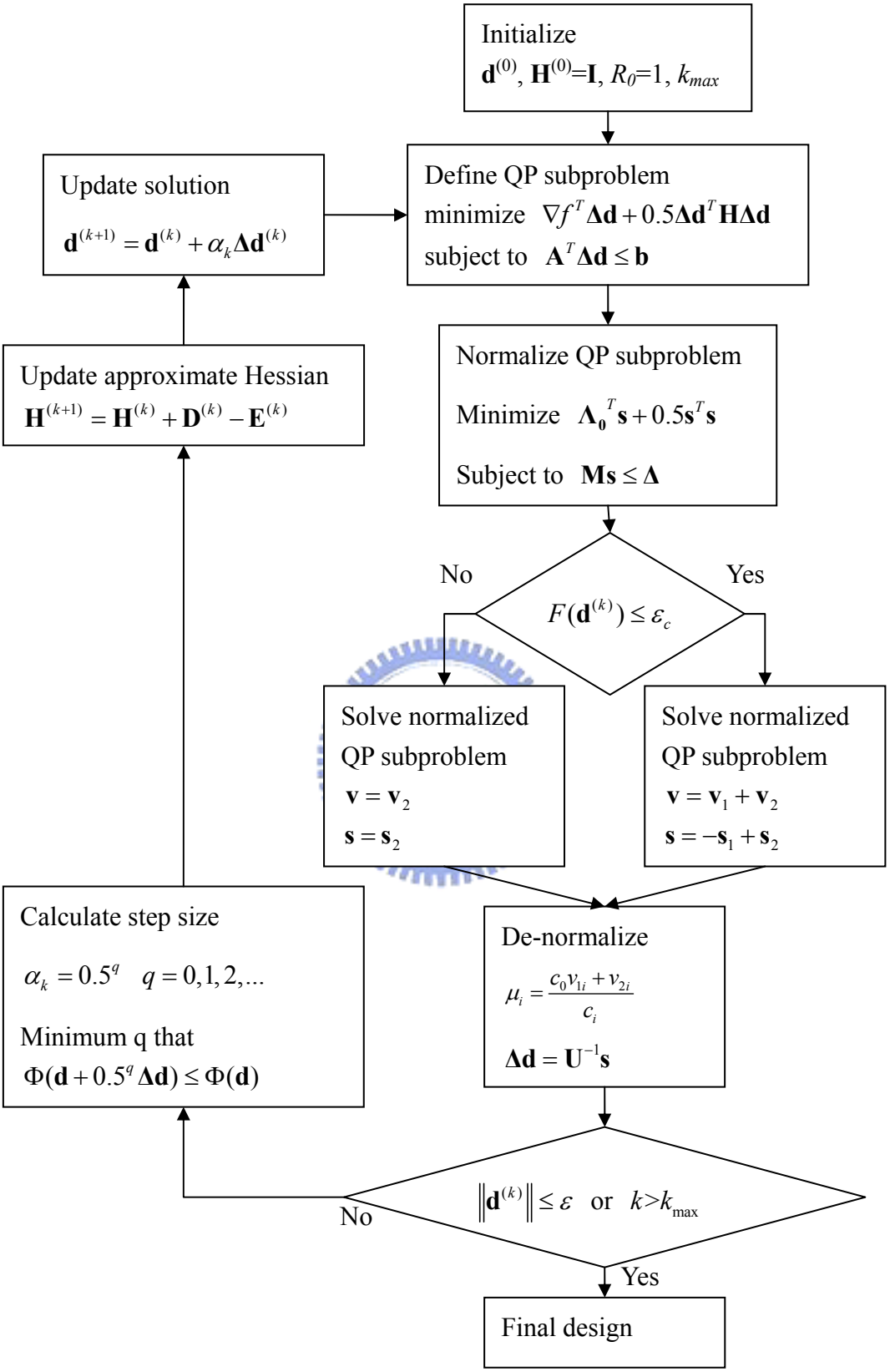
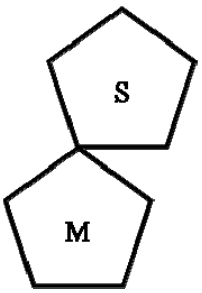
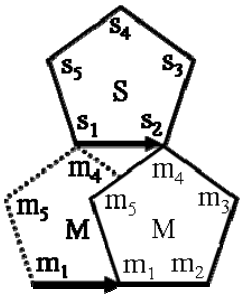


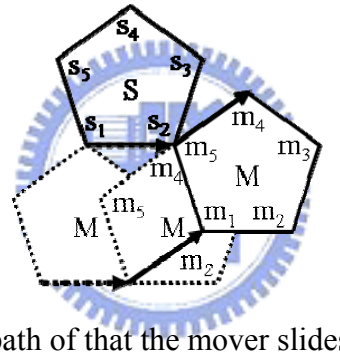
Fig. 3-3 The flowchart of SQP method



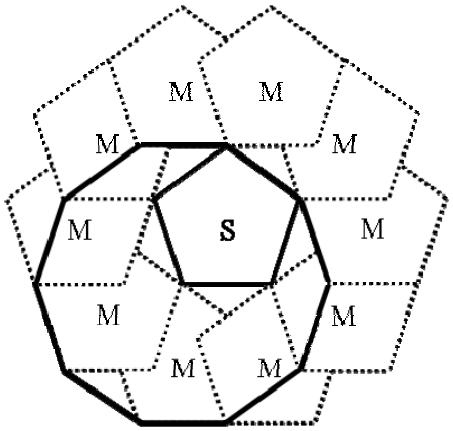
(a) Two objects contact on a point



(b) First path of that the mover slides on the stator

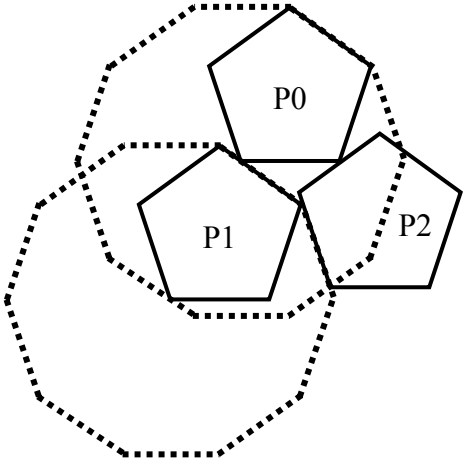


(c) First two path of that the mover slides on the stator

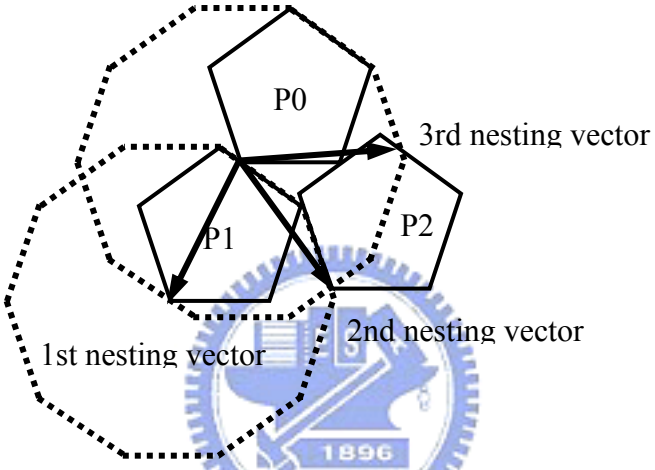


(d) The self-sliding no-fit polygon

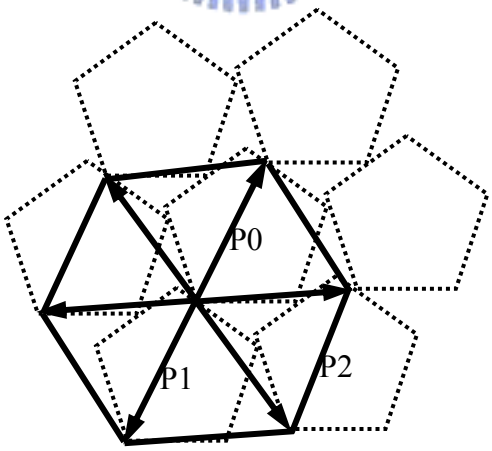
Fig. 3-4 Process of finding the self-sliding no-fit polygon.



(a) The object P1 on the self-sliding no-fit polygon of the object P0



(b) Nesting vectors



(c) The nesting crystal

Fig. 3-5 The process of finding nesting vectors.

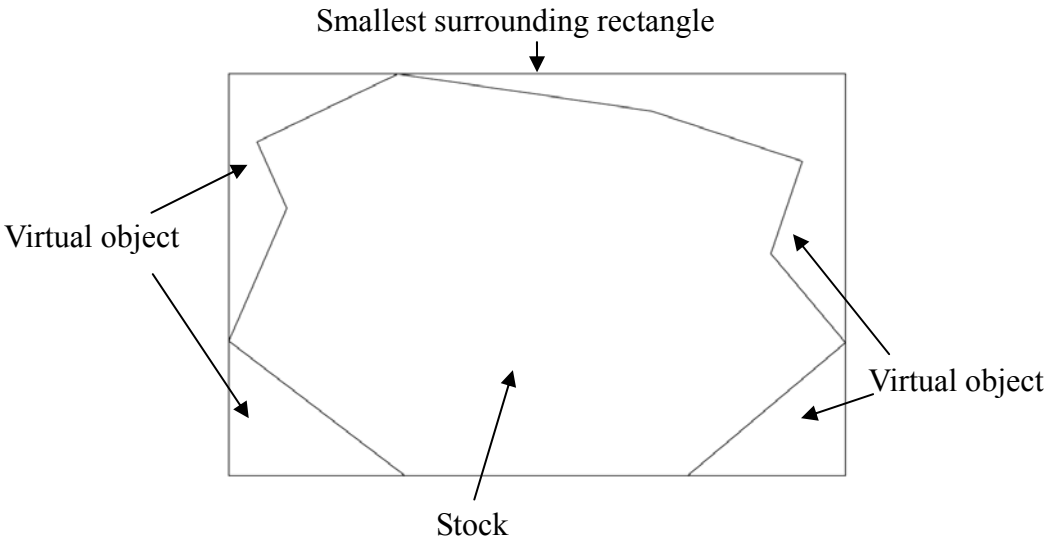


Fig. 3-6 The virtual objects of the irregular stock problem.

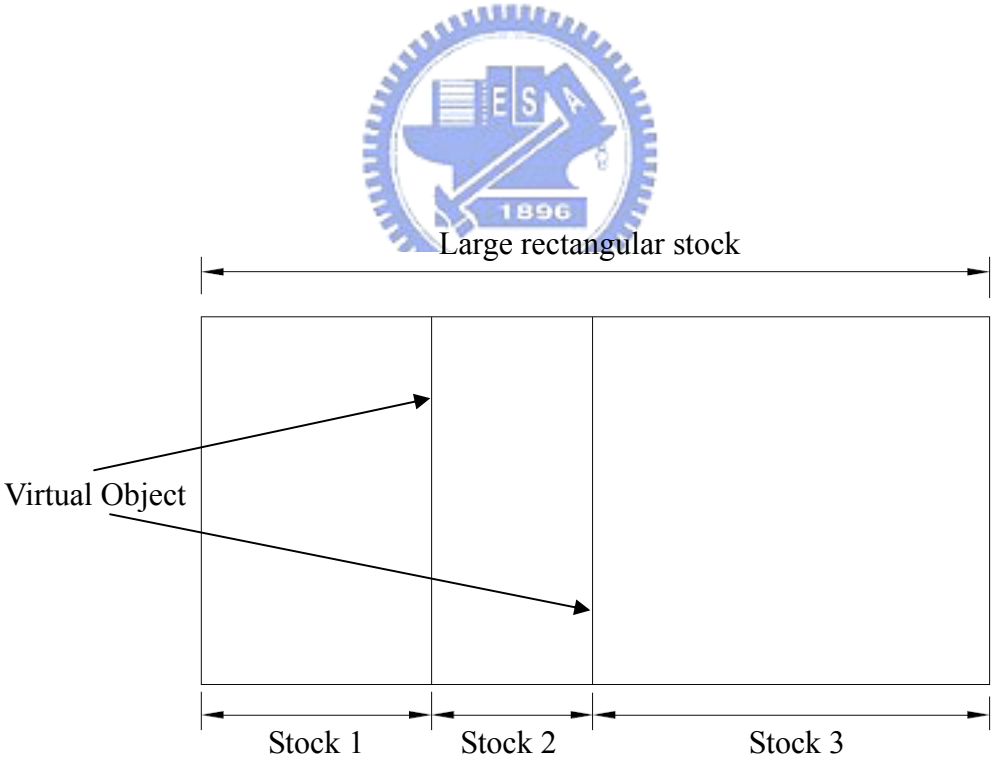


Fig. 3-7 The virtual objects of the multi-stock problem.

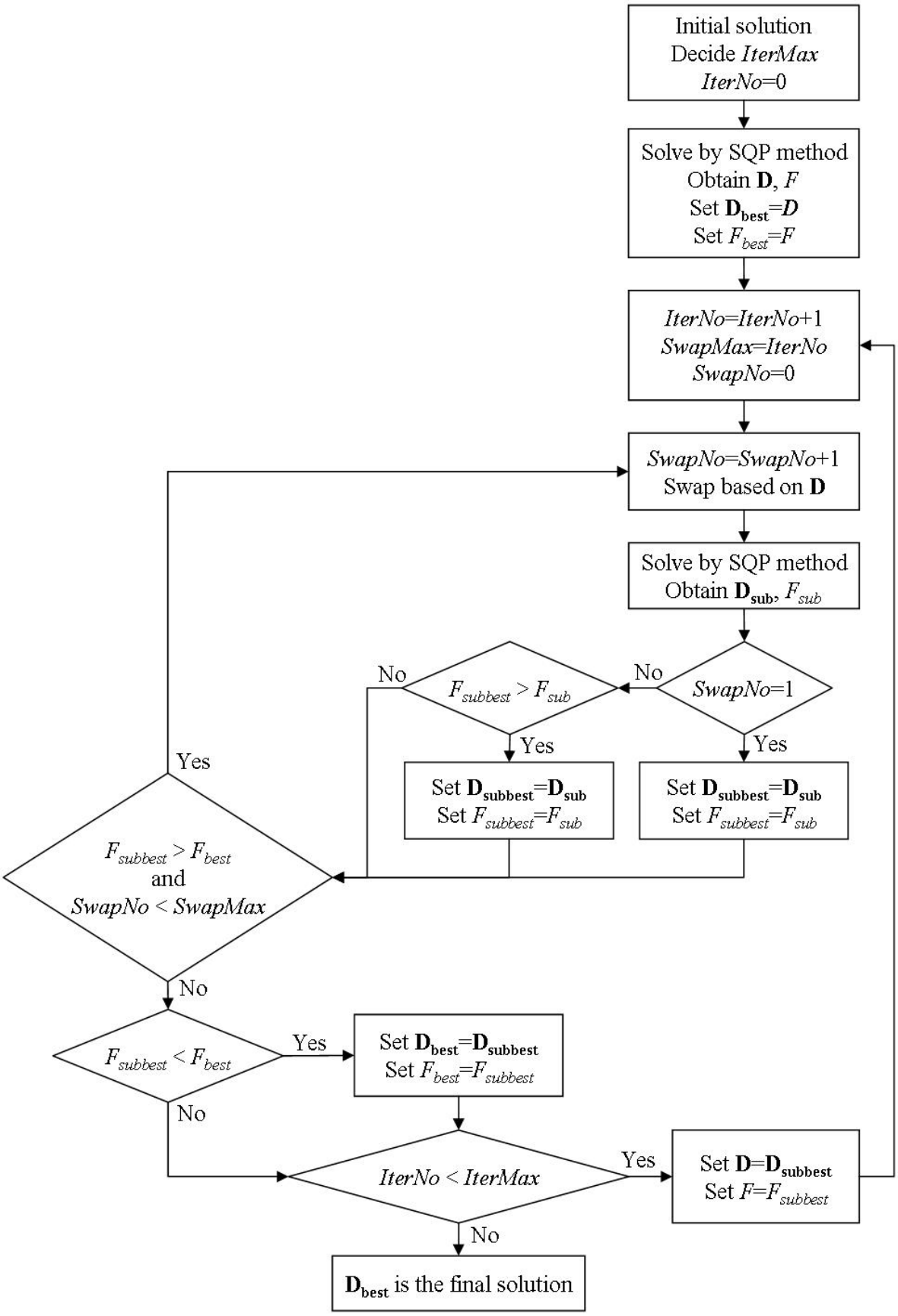


Fig. 3-8 The flowchart of the SQP method with the swap strategy.

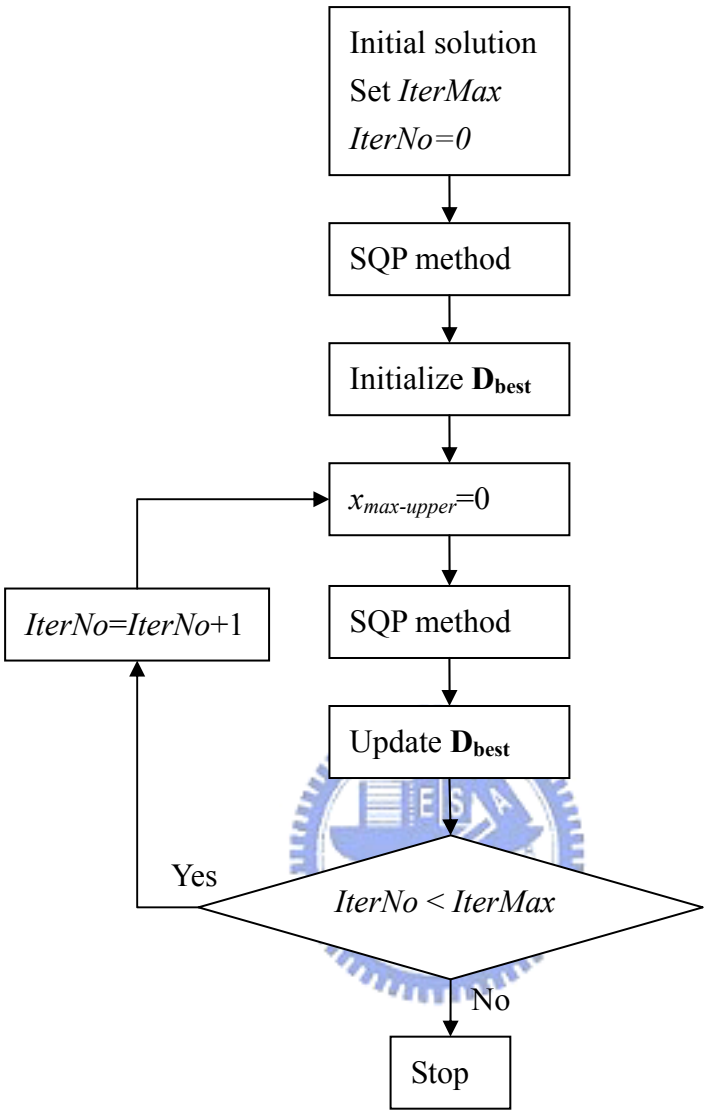


Fig. 3-9 The flowchart of the SQP method with the insert strategy.

CHAPTER 4 EXPERIMENTAL RESULTS

4.1 Combinational problem

The first example used in this section was introduced by Cheng and Rao (1997, 1999, 2000). The profiles of objects that will be nested are shown and numbered as in Fig. 4-1. Objects 2 and 3 have the same profile; objects 7 and 8 also have the same profile.

In the first step, the design variables are the coordinate values (x_i, y_i) and object orientations (θ_i). There are 24 design variables in this case because there are eight objects and every object needs three design variables to represent its position and orientation. The cost function is to minimize the summation distances between objects (as shown in equation (41)), and the 28 constraints are that no two objects may overlap (as shown in equation (43)). This means all objects have to be arranged as close as possible without overlap. Even if there are many constraints in this case, the inactive constraints will be ignored by the active-set strategy in the solving process when using the maximum depth to consider the overlaps.

Deciding the initial solution is another problem in the optimization process, and different initial solutions will lead to different local optimum solutions. The initial solution in the examples of the combinational problem is set by the concept of

initially ignoring the constraints, and findings the best solution in the new unconstrained problem. Then the best solution of the unconstrained problem is set as the initial solution of the constrained problem. Thus, the initial solution is set as that all object positions are on the origin of the global coordinate system. This is because the best solution occurs when all objects overlap on the same position if the constraints are ignored. The object orientations are set as zero, i.e., the original orientation of objects, because they do not affect the cost function but can improve the constraints.

The object positions and orientations are adjusted to improve the constraints in the solving process, resulting in 418 iterations in the SQP solving process in this case. The number of considered constraints during the solving process is shown in Fig. 4-2. At the beginning, 28 constraints are used, i.e., all constraints are considered, but the number of considered constraints is later reduced. For example, there are six pairs of objects that overlap one another in iteration 7. The arranging pattern is shown in Fig. 4-3. This means that there are six considered constraints. The six pairs are object 1 & object 6, object 2 & object 3, object 2 & object 5, object 3 & object 5, object 4 & object 8, and object 7 & object 8. Object 2 and object 3 are almost overlapped completely, and look like one object in the figure. The maximum depths of the first, fifth, and sixth pair are very close to zero, and cannot be observed in the figure. If all

constraints are considered in the solving process, there will be 11704 constraints (28×418). But the summation of the number of considered constraints is 2670 by using the active-set strategy with the maximum depth overlap index. This amounts to a reduction of about 77.19% in this case.

The result of the first step is shown in Table 4-1, and the x- and y-coordinate values are the position of the objects' reference points in the global coordinate system. The arranging pattern of the first step is shown in Fig. 4-4(a), and the objects should be integrated before going into the second step. The profile of the cluster with a highly concave characteristic is shown in Fig. 4-4(b). This concave characteristic may be used in the second step.

The second step uses CNA to nest. The self-sliding no-fit polygon of the cluster has to be found first. Then the positions of the clusters are adjusted to find the optimum nesting vectors as introduced above. The optimum nesting vectors are shown in Table 4-2. The nesting pattern with these nesting vectors has 60 clusters in a 50×50 sheet stock as shown in Fig. 4-5. N_1, N_2, and N_3 are the first, second, and third optimum nesting vectors, respectively. As shown in Table 4-2 and Fig. 4-5, the optimum nesting vectors are not parallel to the X- or Y-axis. Therefore, the nesting pattern can be improved in the third step.

In the third step, the three nesting vectors are aligned to the X- and Y-axis

respectively, as shown in Fig. 4-6. The best case is paralyzing the third nesting vector to the Y-axis, and it is resulting in 66 clusters in a 50×50 stock.

For evaluating the proposed method, three kinds of sheet stocks were used, namely 50×50, 100×100, and 200×200. The cluster in the literature of Cheng and Rao (2000) is shown in Fig. 4-7. Because the Genetic Algorithm is a controlled random method, the improvement step runs three times in every kind of stock with the literature cluster, as shown in Table 4-3(a). The results of the proposed method with these sheet stocks are shown in Table 4-3(b). The comparison between these cases is shown in Table 4-3(c). After integrating the multi-polygon as a single object and nesting by CNA, there are 58 literature clusters in the 50×50 sheet stock, while the proposed method yields 60. Thus, the number of objects is improved by 3.45%. In the 100×100 and 200×200 sheet stock, the number of objects is improved by 5.28% and 4.91% respectively. However, after improving the nesting pattern by the Genetic Algorithm, there are up to 62 clusters in the literature method, while there are 66 clusters in this study after the third step. Thus, the number of objects is improved by 6.45%. The proposed method improves the number of objects in these cases by between 3 and 6 percent. Therefore, the proposed method has better results for rotatable objects and requires less calculation effort.

For testing the efficiency of the methods, this study uses other two cases called

Com_Dagli and Com_Swim. The case information is shown in Appendix A.1. These two cases are modified from the case “Dagli” and “Swim” in the ESICUP website (http://paginas.fe.up.pt/~esicup/tiki-index.php). The clusters are shown in Fig. 4-8. The stock sizes are selected randomly, and the results are shown in Table 4-4 and Table 4-5. The results by using the SQP method and parallism strategy are better than the self-sliding with Genetic Algorithm. Thus, the approach proposed in this study is a good method not only for one special case.

4.2 Multi-polygon problem with a rectangular stock

The object information of the cases used in this section is shown in Appendix A.2. Four cases called Dagli, Swim, Albano, and Shapes2 are used to test the effect of the proposed approach. The first three cases are garment cases and the last one is an artificial case.

The experiment results are shown in Table 4-6. At first, every case is solved by the Swap strategy with the maximum iteration number set as 80. Because the initial solution of the strategy is generated randomly, every case runs 20 times. The results are shown in the row “Swap 80” of Table 4-6. As shown in the table, the results are worse than the results in the literature unless case Swim. Thus, the maximum iteration number is set as 200 in Swap strategy for solving case Dagli, Albano, and Shapes2. Every case runs only 10 times because 200 iteration cost much time. The results are

shown in row “Swap 200” of the table.

For observing easily, the cells of the results are filled with different colors. For case Dagli, Albano, and Shapes2, the results in the literature are compared with results in row Swap 200. If the result of Swap 200 is better than the literature results, the cell of literature result is filled with red. If they are the same, the cell is filled with yellow. If the result of Swap 200 is worse, the cell is filled with green. For case Swim, the literature results are compared with the results in Swap 80, and the color setting is similar as other cases. As shown in Table 4-6, the most results of this study are better than the literature results. Therefore, the Swap strategy is good for the cutting-stock problem. The best arranging patterns of cases are shown in Fig. 4-9. The white parts are objects and the gray part is the unused stock.

Another strategy proposed in this study is the Insert strategy. It is difficult to compare the efficacy of Swap and Insert strategy with fixed iteration number because the number for executing SQP method in Swap 80 is not the same in every run. Thus, the SQP method execution number in every run of the Insert strategy is set the same as that in every run of the Swap strategy respectively. For example, the SQP method execution number in case Dagli run no. 1 of the Swap strategy is 2679, and the SQP method execution number in case Dagli run no. 1 of the Insert strategy is set as 2679, too. The results of the Insert strategy are shown in row “Insert” of Table 4-6. It is

obvious that the results of the Swap strategy are better than the results of the Insert strategy, but the Insert strategy is more stable than the Swap strategy because its deviation is smaller than the Swap strategy's.

4.3 Multi-polygon problem with several rectangular stocks

The object information of the cases used in this section is shown in Appendix A.3. Four cases called Ext_Dagli, Ext_Swim, Ext_Albano, and Ext_Shapes2 are used to test the effect of the proposed approach. These four cases are modified from the cases in the last section, and the difference is the stock number and the stock size.

The maximum iteration number of Swap strategy is set as 80, and there 20 runs in every case. The maximum iteration number of the Insert strategy is depended on the SQP method excitation number in the Swap strategy as describing in the last section.

The results of the Swap and Insert strategy are shown in Table 4-7. The results show that the best result of the Swap strategy is usually better than the best result of the Insert strategy, but the average stock utility rate of the Insert strategy is usually better than the Swap strategy's. It is similar to the single-stock problem that the Insert strategy is more stable than the Swap strategy because its deviation is smaller than the Swap strategy's. The best arrangement patterns of the problem with several rectangular stocks are shown in Fig. 4-10.

4.4 Multi-polygon problem with irregular stocks

The virtual objects in this section are used for multi-stock, irregular stock, and the flaw. The case information is shown in Appendix A.4. Both Swap and Insert strategy are used, and there is only one run for every strategy in every case because the effect of different strategies are compared in the last two section. It is not necessary to compare them again. The results are shown in Table 4-8, and the best arranging patterns are shown in Fig. 4-11. The black part is the flaw on the stock. The results show that the Swap strategy, Insert strategy, and Virtual object strategy are workable for multi-stock problem, irregular stock problem, and flaw stock problem.

4.5 Simplification model

Another effect should be discussed is the effect of simplification model that simplifies the calculation of constraint gradient as shown in section 3.4.2. The original model will calculate all constraint gradients by finite difference method. The case Dagli, Swim, Albano, and Shapes2 in section 4.2 are used to test the effect of the simplification model. Every case has 20 rums for SQP method, and every run has 200 iteration of SQP method. The results are shown in Table 4-9. The results show that the simplification can save more than 95% time of the original model.

Table 4-1 The data of the first step result

Object No.	x coord. value	y coord. value	Orientation (degree)
1	0.305	-2.808	6.922
2	2.723	1.155	-49.388
3	-1.196	-0.911	52.126
4	-0.138	0.985	-3.821
5	1.737	-0.717	-11.513
6	0.973	0.707	5.641
7	0.451	-0.904	92.524
8	1.088	-0.346	106.560

Table 4-2 Nesting vectors

First nesting vector	(1.917, -5.238)
Second nesting vector	(5.586, 1.060)
Third nesting vector	(3.669, 6.298)

Table 4-3 Results — (a) results of CNA with Genetic Algorithm; (b) results of the proposed method; (c) comparing the results.

(a)

The sheet stock size	50×50	100×100	200×200
CNA	58	265	1140
Improving run 1	60	269	1155
Improving run 2	62	272	1150
Improving run 3	62	277	1156

(b)

The sheet stock size		50×50	100×100	200×200
CNA		60	279	1196
Paralleling X-axis	First nesting vector	62	285	1215
	Second nesting vector	61	279	1226
	Third nesting vector	61	276	1216
Paralleling Y-axis	First nesting vector	62	286	1214
	Second nesting vector	62	280	1225
	Third nesting vector	66	284	1216

(c)

The sheet stock size	50×50	100×100	200×200
The best of proposed method	66	286	1226
The best of CNA with GA	62	277	1156
Improvement ratio	6.45%	3.25%	6.06%

Table 4-4 Results of case Com_Dagli (a) results of CNA with Genetic Algorithm; (b) results of the proposed method; (c) comparing the results.

(a)

The sheet stock size	851×1790	1681×1638	649×1490
CNA	1042	1927	647
Improving run 1	1109	2030	687
Improving run 2	1099	2032	695
Improving run 3	1099	2026	687

(b)

The sheet stock size		851×1790	1681×1638	649×1490
CNA		1109	2045	689
Paralleling X-axis	First nesting vector	1131	2096	703
	Second nesting vector	1113	2060	690
	Third nesting vector	1124	2080	695
Paralleling Y-axis	First nesting vector	1146	2092	690
	Second nesting vector	1119	2082	687
	Third nesting vector	1153	2064	707

(c)

The sheet stock size	851×1790	1681×1638	649×1490
The best of proposed method	1153	2096	707
The best of CNA with GA	1109	2032	695
Improvement ratio	3.97%	3.15%	1.73%

Table 4-5 Results of case Com_Swim (a) results of CNA with Genetic Algorithm; (b) results of the proposed method; (c) comparing the results.

(a)

The sheet stock size	144202×124029	48817×71275	69834×95666
CNA	1259	223	451
Improving run 1	1277	230	462
Improving run 2	1282	229	471
Improving run 3	1277	230	468

(b)

The sheet stock size		144202×124029	48817×71275	69834×95666
CNA		1335	238	477
Paralleling X-axis	First nesting vector	1363	236	475
	Second nesting vector	1365	250	486
	Third nesting vector	1334	242	481
Paralleling Y-axis	First nesting vector	1336	243	484
	Second nesting vector	1330	249	484
	Third nesting vector	1337	237	497

(c)

The sheet stock size	144202×124029	48817×71275	69834×95666
The best of proposed method	1365	250	497
The best of CNA with GA	1282	230	471
Improvement ratio	6.47%	8.7%	5.52%

Table 4-6 The results of the problem with a rectangular stock

	Dagli			Swim			Albano			Shapes2		
	Best (%)	Ave. (%)	Dev. (%)	Best (%)	Ave. (%)	Dev. (%)	Best (%)	Ave. (%)	Dev. (%)	Best (%)	Ave. (%)	Dev. (%)
Swap 80	84.21	82.77	0.94	74.08	72.67	0.86	86.49	84.02	1.34	80.23	77.83	1.33
Insert	81.82	80.80	0.82	70.45	69.13	0.67	82.24	80.18	0.96	75.05	73.41	1.09
Swap 200	87.78	86.22	0.83				87.18	84.71	1.28	81.98	79.44	1.74
Egeblad *et al*. (2007), 2DNest	85.98	85.31	0.53	71.53	70.27	0.69	87.44	86.96	0.32	81.21	79.89	1.05
Bouganis and Shanahan (2007), vision	81.00			69.50								
Ratanapan *et al*., (2007)		78.58										
Gomes and Oliveira (2006), GLSHA	85.49	82.99		73.24	71.85		86.41	83.09		81.82	80.24	
Gomes and Oliveira (2006), SAHA	87.15	85.38	1.07	74.37	72.28	0.97	87.43	84.70	1.23	83.60	81.41	0.74
Burke *et al*. (2006), Density 1	83.70			68.40			84.60			79.40		
Poshyanonda and Dagli (2004), GA	84.35											

Table 4-7 The results of the problem with several rectangular stocks.

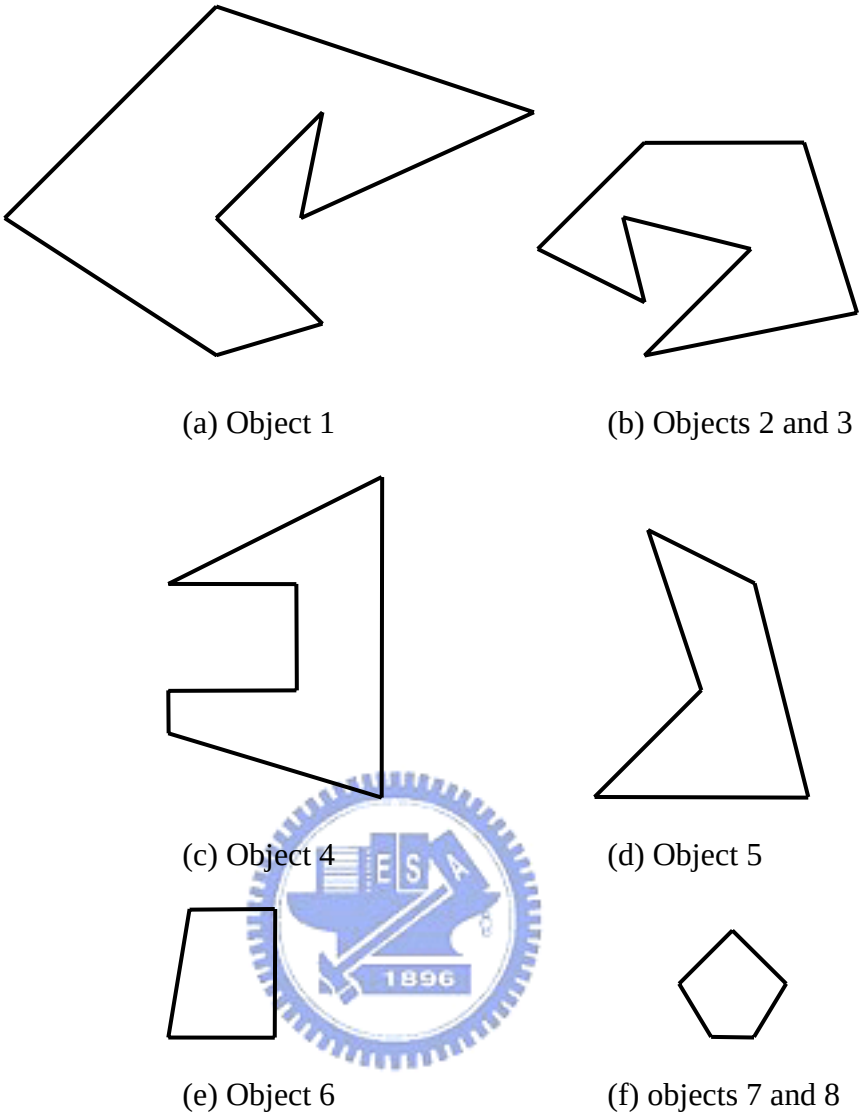
		Ext_Dagli	Ext_Swim	Ext_Albano	Ext_Shapes2
Swap 80	Best	67.48%	61.87%	77.09%	68.67%
	Average	63.99%	57.25%	73.37%	63.42%
	Devaition	2.04%	1.76%	2.46%	2.89%
Insert	Best	78.87%	59.32%	75.34%	67.61%
	Average	77.87%	58.26%	71.88%	65.87%
	Devaition	0.39%	0.80%	1.11%	0.70%

Table 4-8 The results of the problem with irregular stocks.

		Utility
Irr_Dagli	Swap 80	72.97%
	Insert	71.15%
Irr_Swim	Swap 80	39.52%
	Insert	68.99%

Table 4-9 The comparison between the original model and the simplification model.

	Dagli	Swim	Albano	Shapes2
Origin (sec.)	93.75	1707.30	50.80	76.65
Simplification (Sec.)	3.40	17.20	2.05	2.90
Reduction	96.37%	98.99%	95.96%	96.22%



(a) Object 1 (b) Objects 2 and 3

(c) Object 4 (d) Object 5

(e) Object 6 (f) objects 7 and 8

Fig. 4-1 Profiles of objects for cutting.

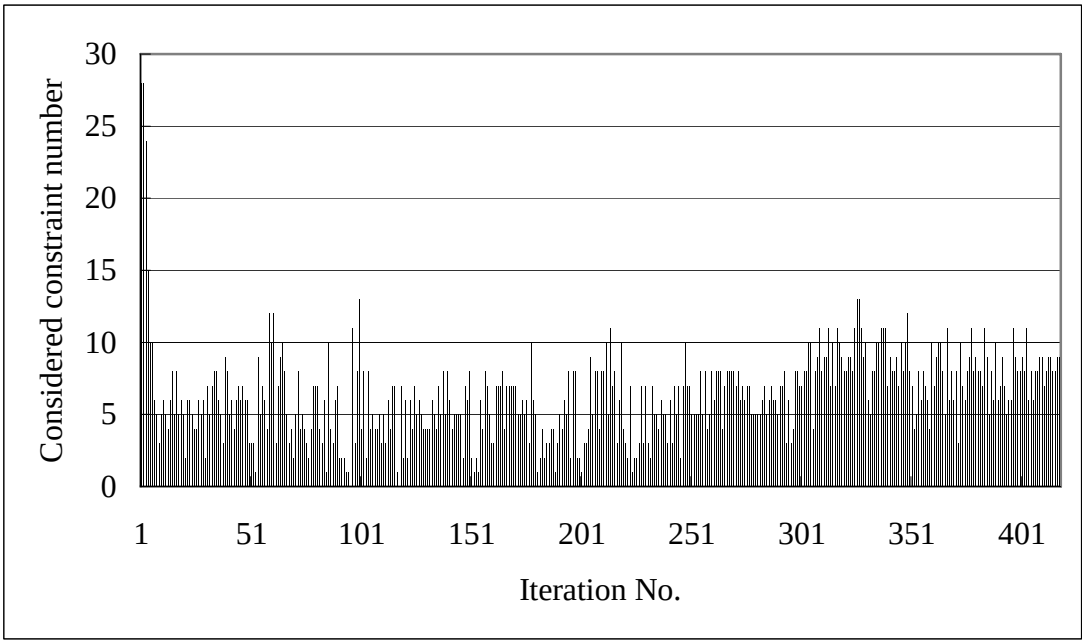


Fig. 4-2 Considered iteration number

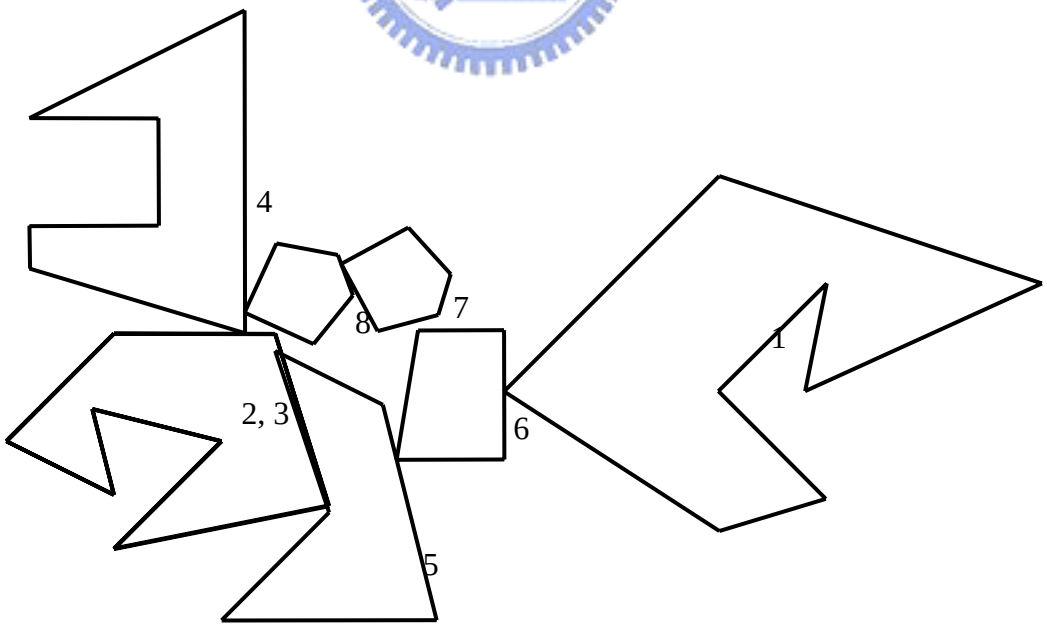
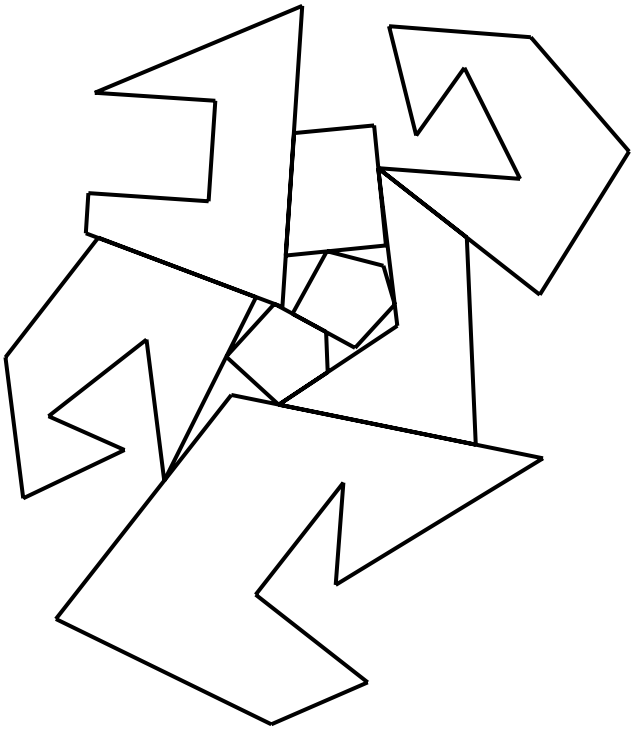
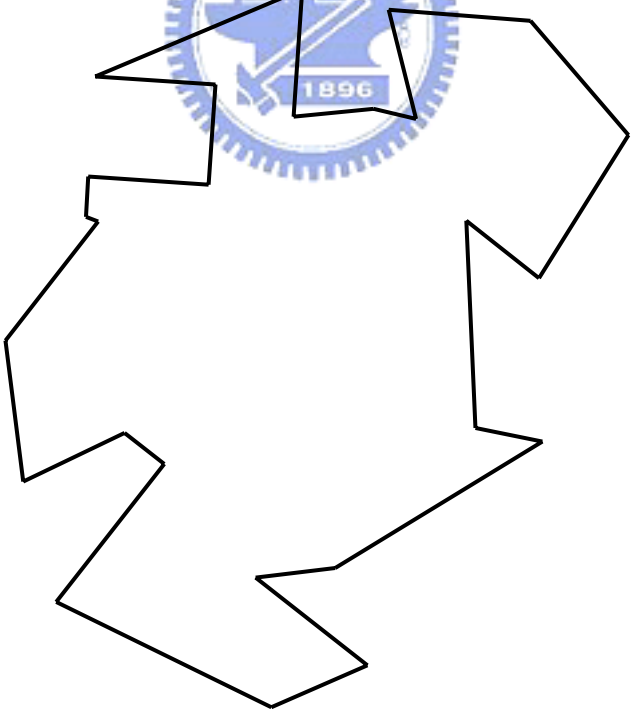


Fig. 4-3 The arranging pattern of iteration 7



(a) Arranging pattern of the first step



(b) Integration object: cluster

Fig. 4-4 Result of the first step.

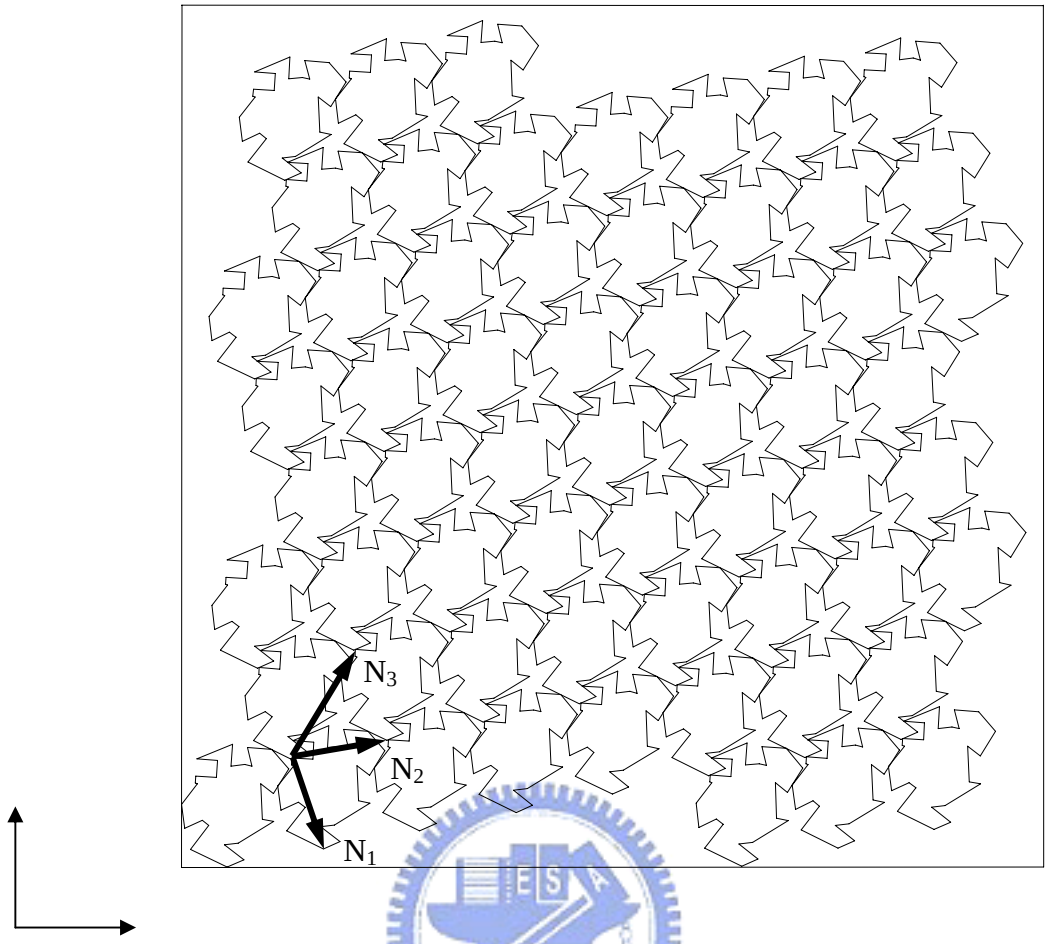
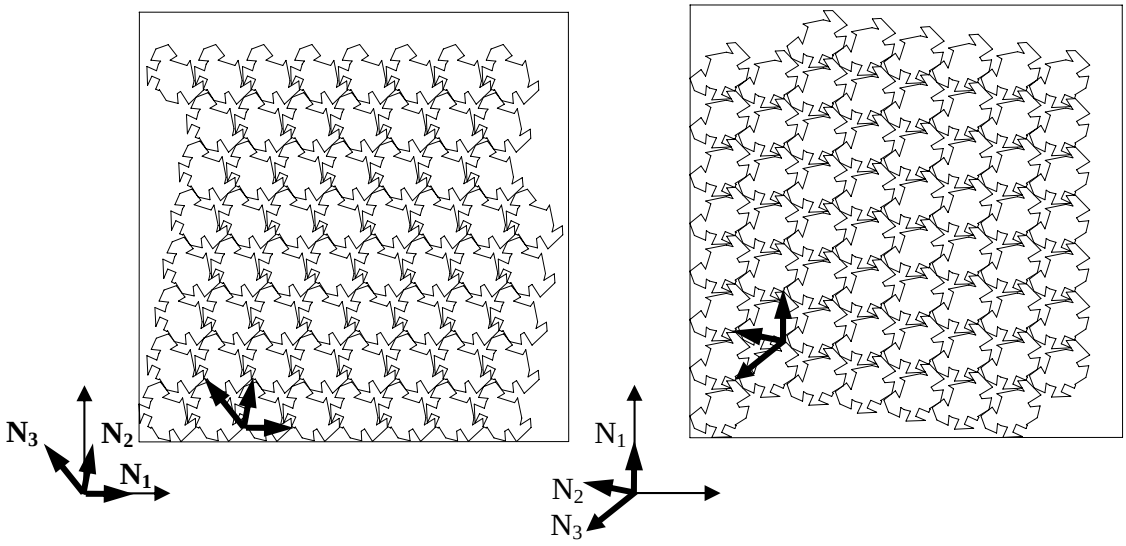
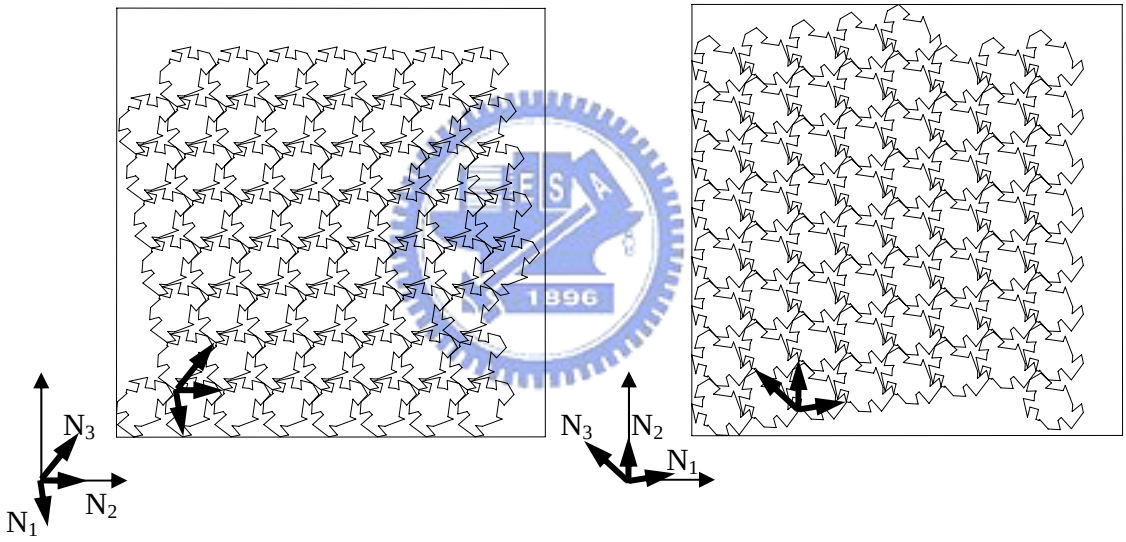


Fig. 4-5 The nesting pattern of the second step on the 50×50 stock

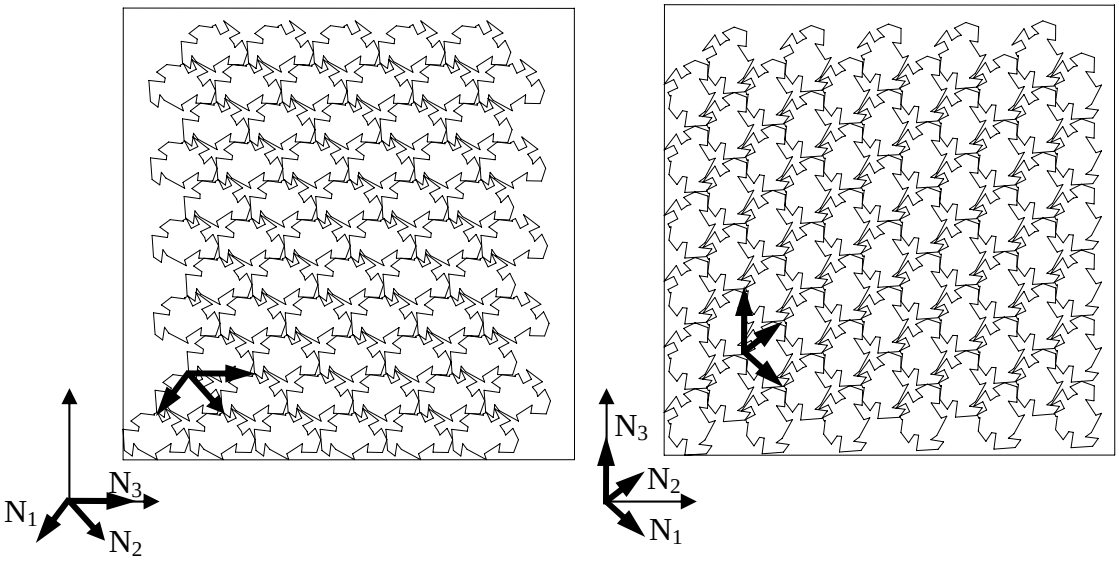


(a) N_1 parallels X-axis (b) N_1 parallels Y-axis



(c) N_2 parallels X-axis (d) N_2 parallels Y-axis

Fig. 4-6 Nesting patterns.



(e) N_3 parallels X-axis (f) N_3 parallels Y-axis

Fig. 4-6. Nesting patterns (continue).

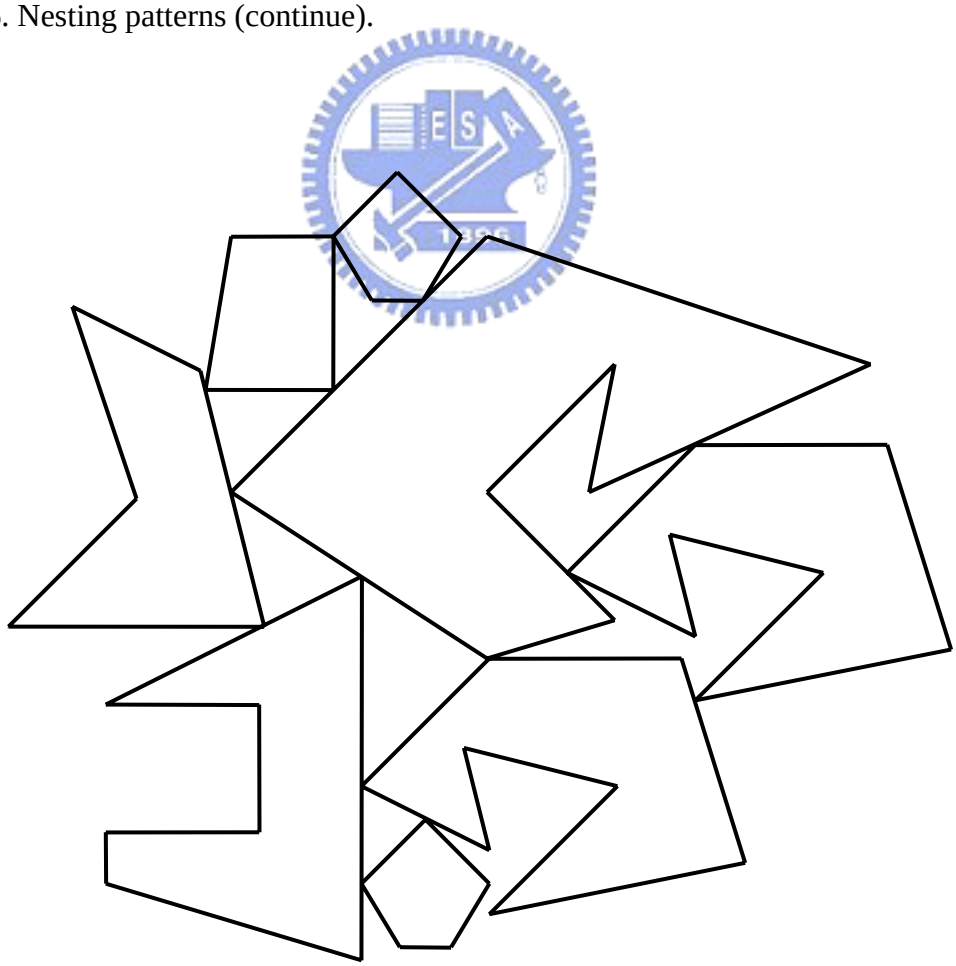
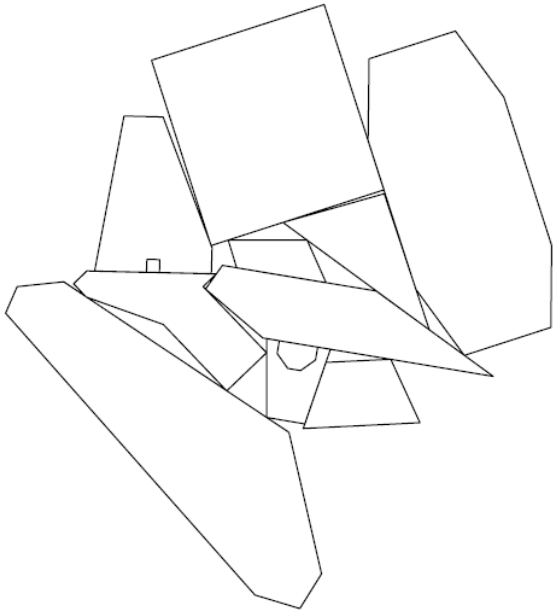
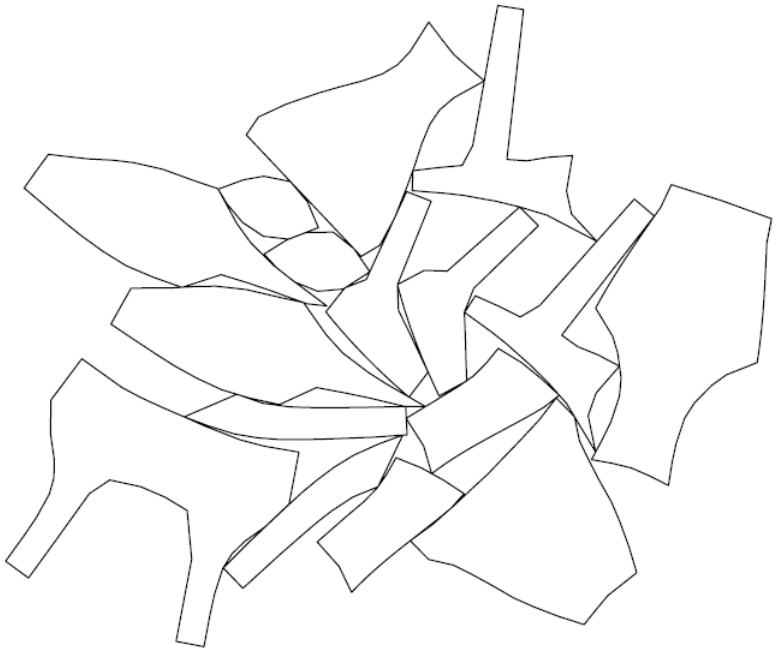


Fig. 4-7 Integration object (cluster) in the literature.

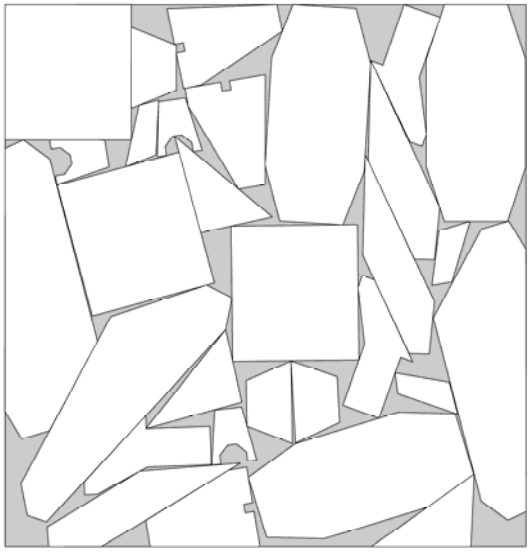


(a) The cluster of case Com_Dagli

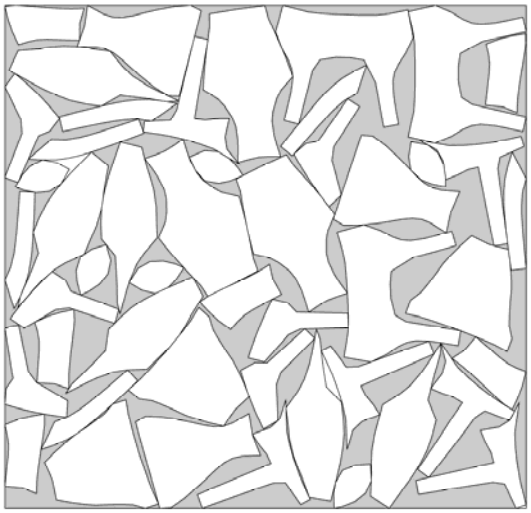


(b) The cluster of case Com_Swim

Fig. 4-8 The clusters of cases.

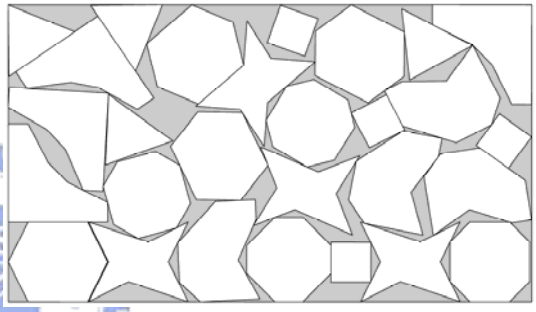


(a) Dagli



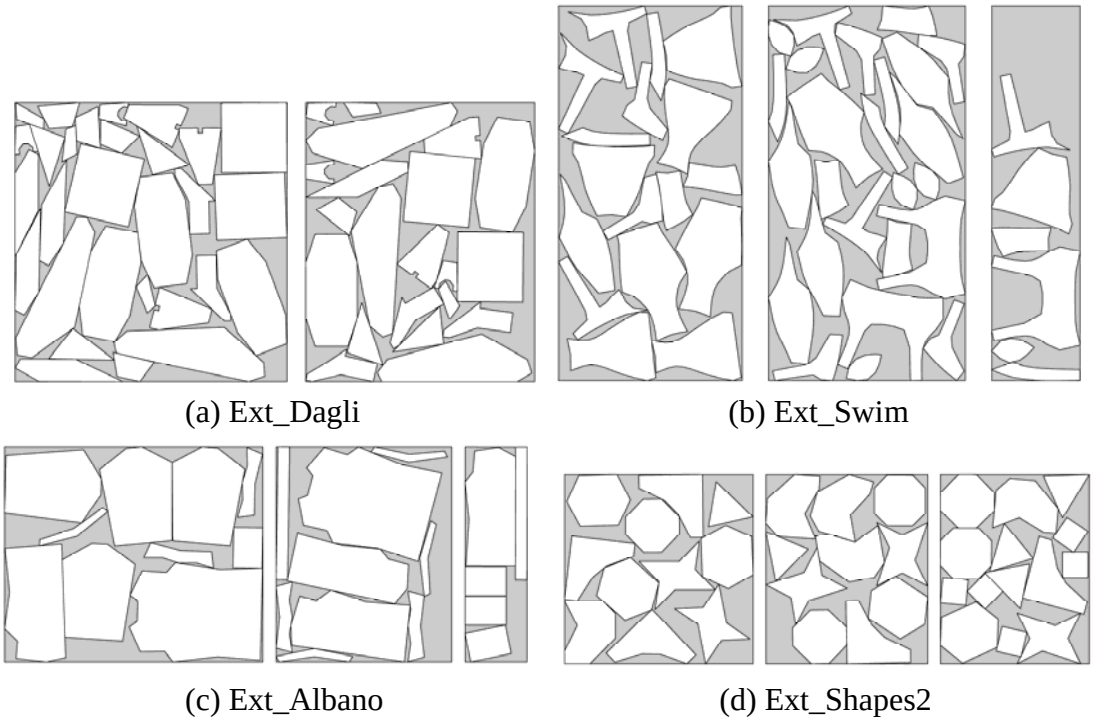
(b) Swim

(c) Albano



(d) Shapes2

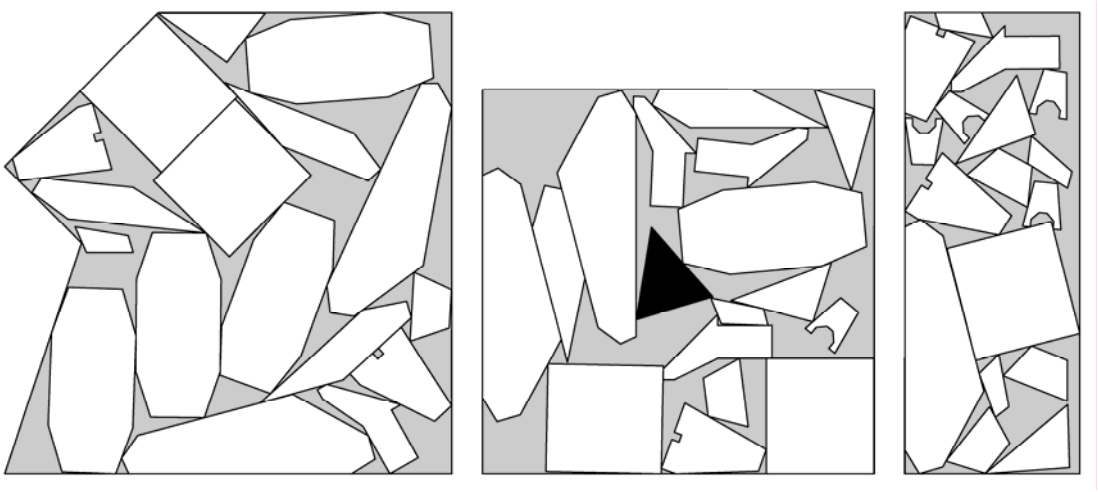
Fig. 4-9 The best arranging pattern of the problem with a rectangular stock.



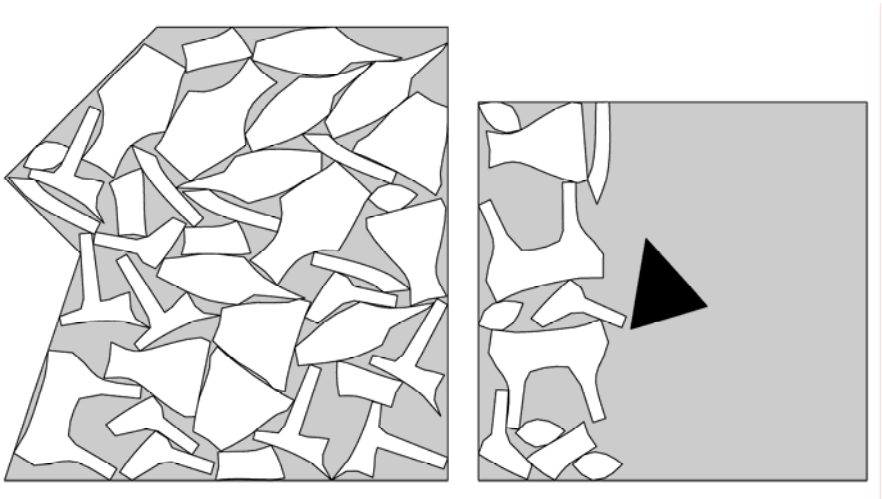
(a) Ext_Dagli (b) Ext_Swim

(c) Ext_Albano (d) Ext_Shapes2

Fig. 4-10 The best arranging pattern of the problem with several rectangular stock.



(a) Irr_Dagli



(b) Irr_Swim

Fig. 4-11 The best arranging pattern of the problem with irregular stocks.

CHAPTER 5 CONCLUSIONS AND FUTURE WORKS

5.1 Conclusions

The cutting-stock problem is considered in many manufacturing industries. According to the statistics by DGBAS (1998) and DGBAS (2003), the cost of stock is about 50% in whole expense of manufacturing industries in Taiwan. Thus, enhancing the stock utility will be helpful for reducing the outgoing of companies in these industries.

Because the cutting-stock is widely considered in many manufacturing industries, it has many different types. This study classifies the cutting-stock problem and focuses four types of the problem. The cutting-stock problem is also discussed in the viewpoint of the optimization problem, and the functions of the solving strategy are obtained.

Therefore, this study proposes:

1. Using the maximum depth as the overlap index instead of real overlap area and being used with the active-set concept can reduce the calculation effort. This is because the constraints will be ignored if the overlap indices are less than zero, i.e., the maximum depths are less than zero and the constraints become inactive.
2. By using the maximum depth as the overlap index, different orientations of

objects is easy to consider.

3. By using the maximum depth as the overlap index, the objects are not coded in binary matrices and they will not deform in different orientations.
4. The cutting-stock problem is formulated into a constrained optimization problem, and a total solution strategy is proposed.
5. Different orientations of objects are easy to consider by setting the ranges of design variables.
6. By formulating the problem into a constrained optimization problem, the famous solving method “SQP method” can be used, and the existed software can be used.
7. In the combinational problem, the nesting pattern is improved easily in the third step of the arranging strategy by aligning the three nesting vector with the X- and Y-axis, respectively. The best one as the final nesting pattern.
8. A global strategy “Swap strategy” is helpful for improving the solution, and it is easy to use because there is no parameter that has to be set.
9. A global strategy “Insert strategy” is helpful for improving the solution, and it is easy to use because there is no parameter that has to be set.
10. The virtual object strategy is workable for the irregular shape of stocks by finding the smallest surrounding rectangle of the irregular stock and filling the virtual objects in the space between them. The problem becomes a rectangular stock

problem with some fixed virtual objects on it.

11. The virtual object strategy is workable for the plurality of the stock by arranging the stocks of the multi-stock problem as a large stock and arranging virtual objects on the large stock at the position of boundaries of original stocks. The problem becomes a single-stock problem with some fixed virtual objects on it.
12. The virtual object strategy is workable for the flaw of the stock by arranging a virtual object on the flaw.
13. The Insert strategy is more stable than the Swap strategy because its deviation of 20 runs is smaller.
14. The Swap strategy is suggested for solving the single-stock problem.
15. The Insert strategy is suggested for solving the multi-stock problem.
16. The simplification model can save much time for calculating the constraint gradients.

5.2 Future works

This study proposes some methods for multi-types of the cutting-stock problem. Although these methods are workable and can obtain good results, some researches can be done in the future.

1. The searching time may be used as the criteria for finishing the whole searching process. The iteration number is used as the criteria in this study, but it is not a

direct sense for user.

2. The pre-set stock utility rate also can be used as the criteria for finishing the whole searching process. If the user only want to find a result that is “good enough” in a short time, this criterion is much suitable.
3. This study considers the overlap of objects and the stock utility. Other properties can be considered in the problem. For example: the relation of the stock thickness and the gaps between objects, the necessary tool number, the tool path, and the cutting path..
4. The methods proposed in this study are suitable for the free rotatable objects. These methods may be modified for solving the problem that the objects should be arranged on some specific orientations.

REFERENCES

Arora, J.S., (1984). An algorithm for optimum structural design without line search, Chapter 20 in *New Directions in Optimum Structural Design*, Atrek, E., Gallagher, R.H., Ragsdell, K.M., and Zienklewiz, O.C., John Wiley and Sons, New York.

Arora, J.S., (1988). IDESIGN software, Optimal Design Laboratory, College of Engineering, The University of Iowa, Iowa City.

Arora, J.S., (2004). *Introduction to Optimum Design*, Second edition. Elsevier/Academic Press, London.

Babu, A.R., and Babu, N.R., (2001). A genetic approach for nesting of 2-D parts in 2-D sheets using genetic and heuristic algorithms, *Computer-Aided Design*, 22, 879-891.

Bennell, J.A., and Dowsland, K.A., (2001). Hybridising tabu search with optimisation techniques for irregular stock cutting, *Management Science*, 47, 8, 1160-1172.

Bouganis, A., and Shanahan, M., (2007). A vision-based intelligent system for packing 2-D irregular shapes, *IEEE Transactions on Automation Science and Engineering*, 4, 382-394.

Burke, E., Hellier, R., Kendall, G., and Whitwell, G., (2006). A new bottom-left-fill heuristic algorithm for the two-dimensional irregular packing problem,

Operations Research, 54, 587-601.

Cheng, S.K., and Rao, K.P., (1997). Quick and precise clustering of arbitrarily shaped flat patterns based on stringy effect, *Computers & Industrial Engineering*, 33, 485-488.

Cheng, S.K., and Rao, K.P., (1999). Concepts of neighborhood and universal compact yield towards achieving best pattern layout, *International Journal of Production Research*, 37, 3643-3658.

Cheng, S.K., and Rao, K.P., (2000). Large-scale nesting of irregular patterns using compact neighborhood algorithm, *Journal of Materials Processing Technology*, 103, 135-140.

Crispin, A., Clay, P., Taylor, G., Bayes, T., and Reedman, D., (2005). Genetic algorithm coding methods for leather nesting, *Applied Intelligence*, 23, 9-20.

DGBAS, (1998). *The Report on 1996 Industry Commerce and Service Census Taiwan-Fukien Area the Republic of China*, DGBAS, Taipei. (主計處，1998，工商及服務業普查報告，主計處，台北)

DGBAS, (2003). *The Report on 2001 Industry Commerce and Service Census Taiwan-Fukien Area the Republic of China*, DGBAS, Taipei. (主計處，2003，工商及服務業普查報告，主計處，台北)

Dowsland, K.A., Vaid, S., and Dowsland, W.B., (2002). An algorithm for polygon

placement using a bottom-left strategy, *European Journal of Operational Research*, 141, 371-381.

Egeblad, J., Nielsen, B.K., and Odgaard, A., (2007). Fast neighborhood search for two- and three-dimensional nesting problems, *European Journal of Operational Research*, 183, 1249-1266.

ESICUP website (http://paginas.fe.up.pt/~esicup/tiki-index.php)

Gomes, A.M., and Oliveira, J.F., (2002). A 2-exchange heuristic for nesting problems, *European Journal of Operational Research*, 141, 359-370.

Gomes, A.M., and Oliveira, J.F., (2006). Solving irregular strip packing problems by hybridising simulated annealing and linear programming, *European Journal of Operational Research*, 171, 811-829.

Huang, Y., Mei, D., Chen, Z., and Hao, D., (2005). Research on an intelligent leather nesting system, *Proceedings of SPIE - Volume 6040*, 60400R1-60400R6.

Koroupi, F., and Loftus, M., (1991). Accommodating diverse shapes within hexagonal pavers, *International Journal of Production Research*, 29, 1507-1519.

Leung, T.W., Chan C.K., and Troutt, M.D., (2003). Application of a mixed simulated annealing-genetic algorithm heuristic for the two-dimensional orthogonal packing problem, *European Journal of Operational Research*, 145, 530-542.

Liao, W.C., (1990). *Integrated Software for Multifunctional Optimization*, Master

Thesis, National Chiao Tung University, Taiwan, R.O.C.

Marques, V.M.M., Bispo, C.F.G., and Sentieiro, J.J.S., (1991). A system for the compactation of two-dimensional irregular shapes based on simulated annealing, *Proceedings of the 1991 International Conference on Industrial Electronics, Control and Instrumentation - IECON '91*, 1911-1916.

Nee, A.Y.C., Seow, K.W., and Long, S.L., (1986). Designing algorithm for nesting irregular shapes with and without boundary constraints, *CIRP Annals*, 35, 107-110.

Poshyanonda, P., and Dagli, C.H., (2004). Genetic neuro-nester, *Journal of Intelligent Manufacturing*, 15, 201-218.

Ratanapan, K., Dagli, C.H., and Grasman, S.E., (2007). An object-based evolutionary algorithm for solving nesting programs, *International Journal of Production Research*, 45, 4, 845-869.

Syu, H., (1996). *Machine Design Handbook*, Chien-Hung, Taipei. (徐灝，1996，機械設計手冊，建宏出版社，台北)

Tay, F.E.H., Chong, T.Y., and Lee, F.C., (2002). Pattern nesting on irregular-shaped stock using Genetic Algorithm, *Engineering Applications of Artificial Intelligence*, 15, 551-558.

Tseng, C.H., (1989). MOST software, Applied Optimal Design Laboratory,

Department of Mechanical Engineering, The National Chiao-Tung University, Taiwan, R.O.C.

Tseng, C.H., Liao, W.C., and Yang, T.C., (1993). *MOST 1.1 User's Manual*. Technical Report No.AODL-93-01, Department of Mechanical Engineering, National Chiao Tung University, Taiwan, R.O.C.

Wu, T.H., Chen, J.F., Low, C., and Tang, P.T., (2003). Nesting of two-dimensional parts in multiple plates using hybrid algorithm, *International Journal of Production Research*, 41, 3883-3900.

Yuping, Z., Shouwei, J., and Chunli, Z., (2005). A very fast simulated re-annealing algorithm for the leather nesting problem, *International Journal of Advanced Manufacturing Technology*, 25, 1113-1118.

Zhang, Y.P., Zhang, C.L., and Jiang, S.W., (2005). An effect approach for leather nesting, *Journal of Software*, 16(2), 316-323.

APPENDIX A CASE INFORMATION

A.1 Combinational problem

Cheng and Rao:

Objects:

No.	Vertices (x, y)
1	(-0.18, -1.77); (0.82, -1.47); (-0.18, -0.47); (0.82, 0.53); (0.62, -0.47); (2.82, 0.53); (-0.18, 1.53); (-2.18, -0.47)
2 & 3	(-0.7, -1.14); (1.3, -0.74); (0.8, 0.86); (-0.7, 0.86); (-1.7, -0.14); (-0.7, -0.64); (-0.9, 0.16); (0.3, -0.14)
4	(-1.32, -0.78); (0.68, -1.38); (0.68, 1.62); (-1.32,0.62); (-0.12, 0.62); (-0.12, -0.38); (-1.32, -0.38)
5	(-1.16, -0.94); (0.84, -0.94); (0.34, 1.06); (-0.66, 1.56); (-0.16, 0.06)
6	(-0.55, -0.58); (0.45, -0.58); (0.45, 0.62); (-0.35, 0.62)
7 & 8	(-0.2, -0.44); (0.2, -0.44); (0.5, 0.06); (0, 0.56); (-0.5, 0.06)

Stocks:

50×50; 100×100; 200×200

Com_Dagli:

Objects:

No.	Vertices (x, y)
1	(-7.125, 1.875); (-6.125, 0.875); (-0.125, -1.125); (4.875, -6.125); (7.875, -3.125); (2.875, 1.875); (3.875, 2.875); (-6.125, 2.875)
2	(-2.5, 8); (-4.5, -4); (-0.5, -4); (-0.5, -3); (0.5, -3); (0.5, -4); (4.5, -4); (4.5, -2); (0.5, 8)
3	(-3.5, 12); (-5.5, 6); (-5.5, -6); (-3.5, -12); (3.5, -12); (5.5, -6); (5.5, 6); (3.5, 12)
4	(-7, 7.5); (-7, -7.5); (7, -7.5); (7, 7.5)
5	(-4.5, -2.5); (4.5, -2.5); (2.5, 2.5); (-2.5, 2.5)
6	(-4, 16.143); (-3, -12.857); (-1, -15.857); (2, -14.857); (7, -4.857); (1, 15.143); (-2, 17.143)
7	(-8.4, 0.4); (-3.4, -2.6); (14.6, -2.6); (4.6, 2.4); (-7.4, 2.4)
8	(-1.5, -4.2); (1.5, -4.2); (2.5, 1.8); (1.5, 1.8); (1.5, 0.8); (0.5, -0.2); (-0.5, -0.2); (-1.5, 0.8); (-1.5, 1.8); (-2.5, 1.8)
9	(-3.75, 0.75); (-2.75, -1.25); (4.25, -1.25); (2.25, 1.75)
10	(-6.333, -2.133); (7.667, -2.133); (-1.333, 4.267);

Stocks:

851×1790; 1681×1638; 649×1490

Com_Swim:

Objects:

No.	Vertices (x, y)
1	(-9.304, 3.188); (-9.354, -0.022); (-9.294, -3.232); (-7.794, -3.412); (-6.714, -3.722); (-5.074, -4.192); (-3.834, -4.512); (-2.504, -4.82188); (-0.724, -5.182); (0.576, -3.632); (1.446, -3.012); (2.806, -2.342); (3.886, -2.082); (5.506, -2.072); (6.666, -2.212); (8.066, -2.452); (7.886, -1.212); (7.776, 0.018); (7.876, 1.248); (8.056, 2.488); (6.656, 2.238); (5.496, 2.098); (3.876, 2.098); (2.796, 2.358); (1.436, 3.018); (0.556, 3.638); (-0.744, 5.178); (-2.524, 4.808); (-3.854, 4.498); (-5.09, 4.168); (-6.734, 3.688); (-7.814, 3.378)
2	(-3.936, 2.733); (-3.856, 1.253); (-3.667, -0.264); (-3.252, -1.891); (-2.146, -1.757); (-1.056, -1.727); (0.074, -1.897); (1.544, -2.237); (3.704, -2.723); (4.114, -0.767); (4.934, 0.773); (3.614, 0.973); (2.614, 1.183); (1.294, 1.523); (-1.216, 2.213); (-2.762, 2.614)
3	(-4.125, 0.623); (-5.815, 0.193); (-7.275, -0.147); (-9.483, -0.603); (-10.601, -1.122); (-7.855, -1.367); (-6.615, -1.527); (-4.685, -1.957); (-3.305, -2.537); (-1.235, -3.727); (0.105, -3.417); (2.244, -2.966); (3.99, -2.378); (5.125, -1.837); (6.395, -1.147); (8.615, -0.107); (8.715, 1.163); (8.795, 2.403); (6.865, 2.703); (5.385, 2.923); (4.265, 3.063); (2.705, 3.173); (1.04, 3.173); (-0.725, 2.913); (-2.525, 2.515)
4	(-6.711, 0.842); (-6.631, -0.878); (-3.621, -0.818); (-1.871, -0.808); (-0.551, -0.828); (0.469, -0.878); (2.049, -1.098); (3.599, -1.638); (6.804, -0.269); (5.379, 0.272); (3.779, 0.774); (2.379, 1.002); (1.139, 1.102); (-0.031, 1.132); (-1.601, 1.102); (-4.581, 0.982)
5	(-1.285, 3.988); (-0.945, 2.559); (-0.515, 1.308); (-0.395, 0.108); (-9.545, -0.402); (-9.525, -1.992); (-0.895, -2.082); (0.405, -2.512); (0.805, -4.012); (1.225, -5.412); (2.425, -5.252); (2.255, -3.272); (2.195, -2.112); (2.215, -0.132); (2.315, 1.348); (2.535, 3.048); (3.602, 6.309); (2.235, 4.528); (0.885, 3.988);

6	(-7.243, -5.072); (-5.945, -5.36); (-4.174, -4.77); (-2.293, -3.9); (-1.083, -3.28); (0.127, -2.62); (1.307, -2.1); (2.347, -1.87); (3.467, -1.94); (5.507, -2.45); (5.357, -1.06); (5.257, -0.01); (5.357, 1.06); (5.507, 2.45); (3.467, 1.94); (2.347, 1.87); (1.307, 2.1); (0.127, 2.62); (-1.083, 3.28); (-2.293, 3.9); (-4.174, 4.771); (-5.945, 5.361); (-7.243, 5.073)
7	(-4.125, 0.036); (-4.165, -1.194); (-4.205, -2.874); (-4.295, -4.504); (-4.525, -6.164); (-2.935, -6.154); (-1.665, -5.344); (-0.605, -4.974); (0.795, -4.544); (2.345, -4.124); (3.535, -3.654); (5.005, -2.984); (6.485, -2.204); (6.335, -0.964); (6.245, 0.056); (6.485, 2.276); (5.005, 3.056); (3.535, 3.726); (2.345, 4.196); (0.795, 4.616); (-0.605, 5.046); (-1.665, 5.416); (-2.935, 6.226); (-4.525, 6.236); (-4.295, 4.576); (-4.205, 2.946); (-4.165, 1.266)
8	(1.313, 2.71); (0.293, 1.56); (-0.627, 0.5); (-2.047, -0.181); (-7.507, -0.271); (-7.467, -1.871); (-1.587, -2.041); (-0.617, -3.031); (1.173, -3.521); (2.033, -1.831); (2.473, -0.871); (3.013, 0.44); (3.475, 1.656); (3.963, 3.14); (2.118, 3.61)
9	(-0.692, 2.401); (-1.713, 1.174); (-1.742, -0.464); (-1.306, -1.654); (-0.476, -3.051); (0.338, -2.345); (1.250, -1.245); (1.849, 0.152); (1.754, 1.694); (0.738, 3.339)
10	(1.422, 7.019); (0.136, 6.234); (-0.894, 5.814); (-1.924, 5.664); (-3.554, 5.824); (-4.724, 6.054); (-6.724, 6.504); (-7.114, 4.824); (-4.424, 4.214); (-2.134, 3.704); (-0.954, 3.444); (0.296, 2.644); (0.656, 0.744); (0.656, -0.876); (0.436, -2.406); (-2.154, -3.826); (-4.444, -4.326); (-7.134, -4.926); (-6.754, -6.606); (-4.754, -6.166); (-3.574, -5.936); (-1.944, -5.786); (-0.914, -5.936); (0.106, -6.366); (1.372, -7.139); (3.24, -7.946); (6.366, -7.246); (5.876, -5.156); (5.656, -3.686); (5.586, -1.476); (5.596, -0.076); (5.596, 1.324); (5.636, 2.804); (5.746, 4.164); (6.396, 7.094); (3.335, 7.824)

Stocks:

1442.02×1240.29; 488.17×712.75; 698.34×956.66

A.2 Multi-polygon problem with a rectangular stock

Dagli:

Objects:

Total object: 30

No.	Amount	Vertices (x, y)
1	3	The same as object 1 of case Com_Dagli shown in A.1.
2	3	The same as object 2 of case Com_Dagli shown in A.1.
3	3	The same as object 3 of case Com_Dagli shown in A.1.
4	3	The same as object 4 of case Com_Dagli shown in A.1.
5	3	The same as object 5 of case Com_Dagli shown in A.1.
6	3	The same as object 6 of case Com_Dagli shown in A.1.
7	3	The same as object 7 of case Com_Dagli shown in A.1.
8	3	The same as object 8 of case Com_Dagli shown in A.1.
9	3	The same as object 9 of case Com_Dagli shown in A.1.
10	3	The same as object 10 of case Com_Dagli shown in A.1.

Stock:

Width: 60

Swim

Object

Total object: 48

No.	Amount	Vertices (x, y)
1	3	The same as object 1 of case Com_Swim shown in A.1.
2	6	The same as object 2 of case Com_Swim shown in A.1.
3	6	The same as object 3 of case Com_Swim shown in A.1.
4	6	The same as object 4 of case Com_Swim shown in A.1.
5	6	The same as object 5 of case Com_Swim shown in A.1.
6	3	The same as object 6 of case Com_Swim shown in A.1.
7	3	The same as object 7 of case Com_Swim shown in A.1.
8	6	The same as object 8 of case Com_Swim shown in A.1.
9	6	The same as object 9 of case Com_Swim shown in A.1.
10	3	The same as object 10 of case Com_Swim shown in A.1.

Stock:

Width: 57.52

Albano:

Objects:

Total objects: 24

No.	Amount	Vertices (x, y)
1	2	(-19.553, -10.44); (-9.893, -9.88); (0.277, -11.3); (2.297, -8.92); (7.787, -9.13); (10.447, -3.63); (8.637, -2.3); (8.637, 2.3); (10.447, 3.63); (7.787, 9.13); (2.297, 8.92); (0.277, 11.3); (-9.893, 9.88); (-19.553, 10.44)
2	2	(-15.17, -1.305); (15.17, -1.305); (15.17, 1.305); (-15.17, 1.305)
3	4	(-13.147, -6.57); (4.463, -8.3); (8.683, -1.8); (8.683, 1.8); (4.463, 8.3); (-13.147, 6.57)
4	4	(-7.96, -1.123); (0, 0.067); (7.96, -1.123); (8.7, 0.127); (0, 1.927); (-8.7, 0.127)
5	4	(-8, -1.44); (-3.89, -0.79); (0, -1.44); (3.89, -0.79); (8, -1.44); (7, 2.24); (0, 1.42); (-7, 2.24)
6	4	(-4.68, -3.295); (4.68, -3.295); (4.68, 3.295); (-4.68, 3.295)
7	2	(-15.743, -4.774); (-5.643, -4.074); (2.608, -5.504); (5.558, -2.624); (9.428, -3.094); (10.458, 3.756); (9.638, 8.156); (-16.303, 8.156)
8	2	(-16.194, -7.03); (8.796, -7.03); (10.856, -3.16); (10.026, 2.31); (5.286, 2.64); (3.006, 4.49); (-5.584, 3.56); (-16.194, 4.22)

Stock

Width: 49

Shapes2

Objects:

Total objects: 28

No.	Amount	Vertices (x, y)
1	4	(-2, -1.5); (0, -2.5); (2, -1.5); (2, 1.5); (0, 2.5); (-2, 1.5)
2	4	(-1.25, -2.5); (1.75, -2.5); (0.75, -0.5); (1.75, 1.5); (1.75, 2.5); (-0.25, 2.5); (-2.25, 0.5); (-2.25, -1.5)
3	4	(-1, -2); (1, -2); (2, -1); (2, 1); (1, 2); (-1, 2); (-2, 1); (-2, -1)
4	4	(-2, -2.5); (0, -1.5); (2, -2.5); (1, -0.5); (2, 2.5); (0, 1.5); (-2, 2.5); (-1, 0.5)
5	4	(-2.714, -2.286); (2.286, -2.286); (2.286, 2.714); (1.286, 2.714); (0.286, 0.714); (-0.714, -0.286); (-2.714, -1.286)
6	4	(0, -2); (2, 1); (-2, 1)
7	4	(-1, -1); (1, -1); (1, 1); (-1, 1)

Stock:

Width: 15

A.3 Multi-polygon problem with several rectangular stocks

Objects:

The vertices of objects in Ext_Dagli are the same as vertices of objects in Dagli in A.2. The vertices of objects in Ext_Swim, Ext_Albano, and Ext_Shapes2 are the same as vertices of objects in Swim, Albano, and Shapes2 in A.2 respectively.

Case	Ext_Dagli	Ext_Swim	Ext_Albano	Ext_Shapes2
No. & amount	No. 1: 5 No. 2: 5 No. 3: 5 No. 4: 5 No. 5: 5 No. 6: 5 No. 7: 5 No. 8: 5 No. 9: 5 No. 10: 5	No. 1: 3 No. 2: 6 No. 3: 6 No. 4: 6 No. 5: 6 No. 6: 3 No. 7: 3 No. 8: 6 No. 9: 6 No. 10: 3	No. 1: 2 No. 2: 2 No. 3: 4 No. 4: 4 No. 5: 4 No. 6: 4 No. 7: 2 No. 8: 2	No. 1: 5 No. 2: 5 No. 3: 5 No. 4: 5 No. 5: 5 No. 6: 5 No. 7: 5
Total	50	48	24	35

Stocks:

Case	Ext_Dagli	Ext_Swim	Ext_Albano	Ext_Shapes2
No. 1	60×58.2	57.52×28	49×58.8	15×14.85
No. 2	60×51	57.52×30	49×40	15×12.75
No. 3	60× infinite	57.2× infinite	49× infinite	15× infinite

A.4 Multi-polygon problem with irregular stocks

Objects:

Objects of cases Ext_Dagli and Ext_Swim are used and the object information is the same with they are in A. 3. The cases are called Irr_Gagli and Irr_Swim here.

Stocks

No.	1	2	3
Description	Vertices: (0, 0) (58.2, 0) (58.2, 60) (20, 60) (0 ,40) (10, 30)	Rectangle 50 × 51 with flaw (20, 20) (30, 23) (22, 32)	Regular 60 × infinite