ตัวแบบจำนวนเต็มผสมสำหรับการจัดตารางการผลิตบรรจุภัณฑ์แก้ว

นายชวลิต บุญชม

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต

สาขาวิชาคณิตศาสตร์ประยุกต์และวิทยาการคณนา ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์

คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

ปีการศึกษา 2555

MIXED INTEGER MODEL FOR GLASS CONTAINER PRODUCTION SCHEDULING

Mr. Chaowalit Bunchom

A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Science Program in Applied Mathematics and Computational Science

Department of Mathematics and Computer Science,

Faculty of Science, Chulalongkorn University

Academic Year 2012

Thesis Title	MIXED INTEGER MODEL FOR GLASS CONTAINER PRODUCTION SCHEDULING
By	Mr. Chaowalit Bunchom
Field of Study	Applied Mathematics and Computational Science
Thesis Advisor	Assistant Professor Krung Sinapiromsaran, Ph.D.

Accepted by the Faculty of Science, Chulalongkorn University in Partial Fulfillment of the Requirements for the Master's Degree

……………………………………………………… Dean of the Faculty of Science

(Professor Supot Hannongbua, Dr.rer.nat.)

THESIS COMMITTEE

……………………………………………………… Chairman

(Boonyarit Intiyot, Ph.D.)

……………………………………………………… Thesis Advisor

(Assistant Professor Krung Sinapiromsaran, Ph.D.)

……………………………………………………… Examiner

(Phantipa Thipwipatpotjana, Ph.D.)

……………………………………………………… External Examiner

(Pichanon Suwannatada, Ph.D.)

ชวลิต บุญชม : ตัวแบบจำนวนเต็มผสมสำหรับการจัดตารางการผลิตบรรจุภัณฑ์แก้ว. (MIXED INTEGER MODEL FOR GLASS CONTAINER PRODUCTION SCHEDULING) อ.ที่ปรึกษาวิทยานิพนธ์หลัก: ผศ.ดร.กรุง สินอภิรมย์สราญ, 45 หน้า.

วิทยานิพนธ์นี้นำเสนอการวางแผนและการจัดตารางการผลิตในอุตสาหกรรมบรรจุภัณฑ์แก้ว โดยที่เตาหลอมแก้วจะกระจายน้ำแก้วไปสู่เครื่องจักรขนานที่ไม่สัมพันธ์กันเพื่อขึ้นรูปบรรจุภัณฑ์แก้ว เวลาในการปรับตั้งเครื่องจักรนั้นไม่เป็นอิสระกับงานและมีค่าใช้จ่ายเกิดขึ้นเมื่อมีการเปลี่ยนงาน งานวิจัยนี้แสดงตัวแบบจำนวนเต็มผสมโดยใช้กำหนดการเป้าหมายสำหรับการหาค่าต่ำสุดของเวลาในการปรับตั้งเครื่องจักรและเวลาปิดงาน ตัวแบบทางคณิตศาสตร์ของโปรแกรม SAGE ถูกสร้างขึ้นเพื่อใช้หาคำตอบที่เหมาะที่สุด โดยประสิทธิภาพของตัวแบบจำนวนเต็มผสมจะถูกเปรียบเทียบกับการจัดตารางการผลิตของบริษัทในประเทศไทย ซึ่งพบว่าผลการคำนวณเวลาในการปรับตั้งเครื่องจักรของตัวแบบจำนวนเต็มผสมโดยใช้กำหนดการเป้าหมายทำได้ดีกว่าการจัดตารางการผลิตจริงที่เคยทำมา

ภาควิชา คณิตศาสตร์และวิทยาการคอมพิวเตอร์ ลายมือชื่อนิสิต

ลายมือชื่อ อ.ที่ปรึกษาวิทยานิพนธ์หลัก

สาขาวิชา คณิตศาสตร์ประยุกต์และวิทยาการคณนา

ปีการศึกษา 2555

5471947823: MAJOR APPLIED MATHEMATICS AND COMPUTATIONAL SCIENCE KEYWORDS: PRODUCTION PLANNING / SCHEDULING / MIXED INTEGER PROGRAMMING / GLASS CONTAINER / MAKESPAN / SAGE

CHAOWALIT BUNCHOM: MIXED INTEGER MODEL FOR GLASS CONTAINER PRODUCTION SCHEDULING. ADVISOR: ASST. PROF. KRUNG SINAPIROMSARAN, Ph.D., 45 pp.

This thesis proposes the production planning and scheduling mathematical programming model for the glass container industry. A furnace distributes glass to a set of parallel machines for forming products, where machine-dependent setup time and cost are incurred for switching overs from one product to another. This thesis states the formulation of a mixed-integer programming model using a goal programming for minimizing the setup time and makespan. The SAGE mathematical model was constructed and solved for the optimal solution. The performances of the mixed-integer programming model are compared with the current production scheduling from the company in Thailand. The computational results exhibit better setup time than the current plan.

Department: Mathematics and Computer Science

Student's Signature ....

Advisor's Signature ....

Field of Study: Applied Mathematics and Computational Science

Academic Year: 2012

ACKNOWLEDGEMENTS

I would like to express my sincere thanks to my thesis advisor, Assistant Professor Krung Sinapiromsaran for giving me the suggestion and constant encouragement throughout the course of this research. I am most grateful for his teaching and advice, not only the research methodologies but also many other methodologies in life.

I am grateful to Applied Mathematics and Computational Science, Department of Mathematics and Computer Science, Faculty of Science, Chulalongkorn University and to the company Bangkok Glass for financial and technical support.

Finally, I most gratefully acknowledge my parents and my friends for all their support throughout the period of this research.

CONTENTS

Page

Page

Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

1.1 Motivation

Nowadays, glass containers are widely used around the world due to its transparency, alkaline properties and acidic resistance. Moreover, they are recyclable. Glass container productions have high requested orders every month, which cause a lot of setup cost and manpower cost. Therefore, optimization techniques are needed for organizing available resources by ordering a sequence of operations of orders assigned to each resource.

Many glass manufacturers still use planning staff to schedule the production planning. However, the optimal solution may not be obtained. There are many researches on scheduling problem with the machine-dependent setup time in the glass container industry that use optimization techniques to find an optimal solution. Therefore the problem needs to be formulated as a mixed-integer mathematical programming model, so optimization techniques can be used to help the production planning staff for scheduling the glass production process. Next, the glass container production process will be described for understanding the work flow of the problem.

1.2 Glass container production process

A glass container production process composes of two sections: the hot zone and the cold zone. The hot zone handles furnaces, annealing ovens or lehrs, and forming machines. The cold zone contains product inspection units and packaging equipment units, see Figure 1.1. However, a glass container production process contains three main sub-processes. They are glass mixing and melting (MM), glass containers manufacturing-forming (GF) and palletizing which can be seen in Figure 1.2.

Figure 1.1: A glass container production process

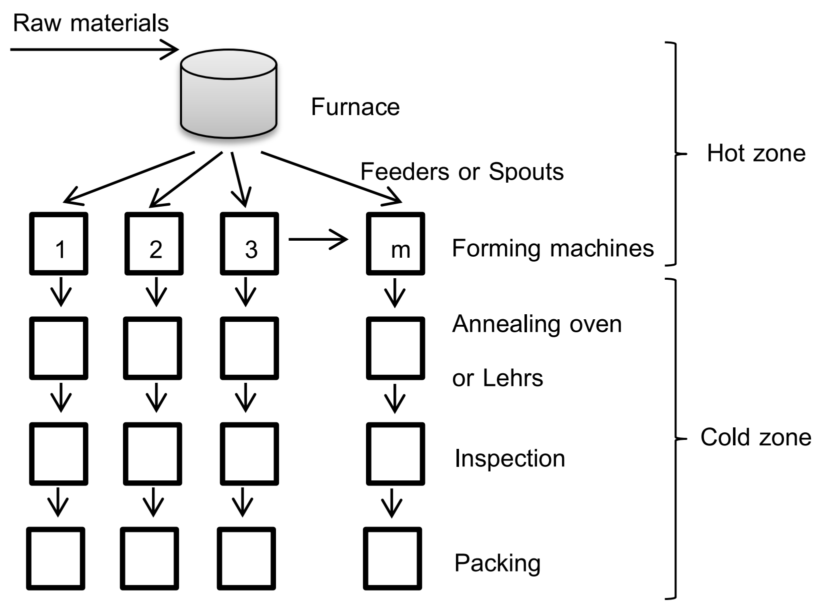
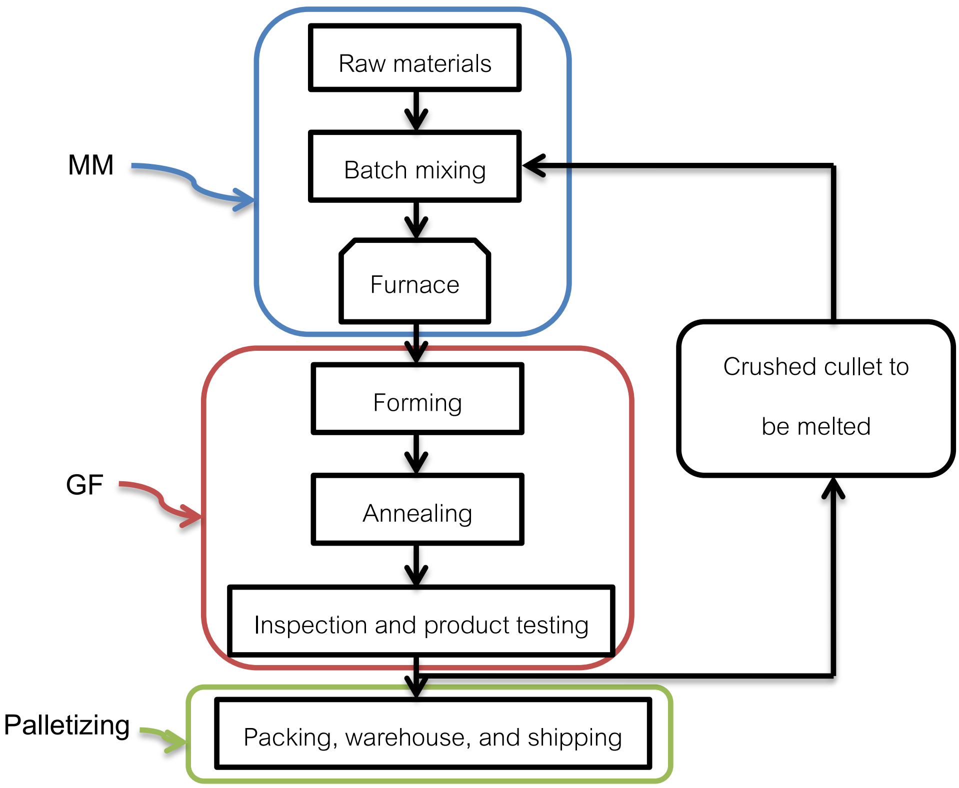


Figure 1.2: The sub-processes of a glass container production process



1.2.1 Raw materials and mixing process

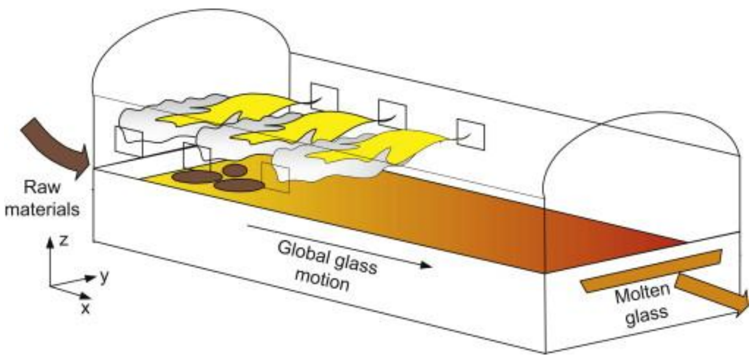
Raw materials (silica sand, feldspar, dolomite, soda ash and limestone) are fed separately into a storage site before mixing with treated cullet processed in the cullet treatment process. The cullet treatment process starts by crushing used bottles. Then, unwanted metal, plastic and paper are filtered out of the glass waste before being mixed with raw materials and other components in a batch mixing. Raw materials are weighted according to a required glass formulae and then mixed with the treated cullet proportionally. The process of melting is applied after obtaining the appropriate mixture of glass production.

1.2.2 Melting process

The mixture is melted at temperature of 1,600 Celsius in a glass furnace which can be seen in Figure 1.3. Then, glass containers are formed by molten raw materials in heat which is generated by fuel oil burners or natural gas or electric booster. In addition, Salt cake, Selenium, Coke dust, and Iron oxide are applied for the required properties and colours. If cullet is used an increasingly large proportion of glass batch, up to 98%, then it can save energy consumption, the main cost in glass container industries.

Figure 1.3: A glass furnace

Source: Olivier Auchet, et al., 2008



The molten glass is transported through the fining, and conditioning zone to the spout. When a molten glass moves to the end of the spout, the glass temperature must be within the required tolerance to make the forming of the bottle possible. The batch material takes about one day to pass through the melting state. The unit of furnace output is a ton per day. The furnace outputs range from 100 tons per day to over 600 tons per day.

1.2.3 Forming process

Molten glass moves through the furnace throat where the temperature is controlled between 1,000 – 1,300 Celsius before it is transported into the parallel forming machine which is shown in Figure 1.4. The molten glass is cut into the cylinder shape which is called 'a gob' according to the weight and design of each bottle. The forming machine has several characteristics; mold sizes and process types, which restrict the set of bottles. Each forming machine can only process one order at a time. There are three forming types; blow and blow (BB) for narrow-neck bottles, press and blow (PB) for wide mouth bottles, and narrow neck press and blow (NNPB) for the light weight bottles. Since the production rate of each machine is not the same which implies the gob speeds for each forming machine is also different. The production rate can be calculated by the formula:

$$PR_{ik} = \frac{GW_i \times GS_{ik} \times 1440}{1000000},$$

where P Set of jobs,

MC Set of machines,

PR_{ik} Production rate of the job i at the machine k for $i \in P$ and $k \in MC$ (tons per day),

GW_i One unit of the job i for $i \in P$ (grams) and

GS_{ik} Speed of the job i at the machine k for $i \in P$ and $k \in MC$ (units per minute).

For example, if we set $P = \{A\}$ and $MC = \{1,2\}$ and gob speeds of each forming machine is 100 and 200 units per minute. Assume that one unit of gob of job A weighs 165 grams. Then,The constant 1440 is the total minutes of one day calculated. The constant 1000000 is used to convert the gram unit to the ton unit. Therefore,

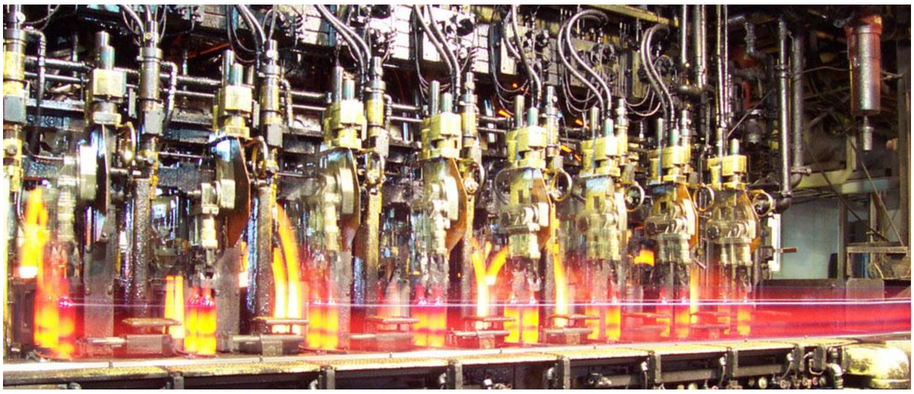
$$PR_{A1} = \frac{165 \times 100 \times 1440}{1000000} = 23.76 \text{ tons per day}$$

$$PR_{A2} = \frac{165 \times 200 \times 1440}{1000000} = 47.52 \text{ tons per day}$$

If the marketing requirement of job A is 500 tons then the processing times of job A for each forming machine are $\frac{500}{23.76} = 21.04$ days and $\frac{500}{47.52} = 10.52$ days, respectively.

Figure 1.4: A glass container forming machine

Source: http://www.recycleglass.co.nz(5th March 2013)



1.2.4 Annealing process

Annealing process is a process of formed bottles which are transported into the annealing oven or lehr for reducing stress, see Figure 1.5. During the move from the forming machine to the oven, the temperature of each bottle will drop. The annealing oven will raise the temperature of the containers to approximately 540 Celsius, holding for specified minutes and then cooling at a consistent rate to remove the stress from predetermined wall thickness.

Figure 1.5: The glass bottles are transporting into the annealing oven.

Source: http://globalpackage.net(5th March 2013)

1.2.5 Quality control and inspection process

After moving out of the annealing process, glass bottles are transported to the inspection process. The inspection process consists of machine check, laboratory check and visual check to guarantee that the durability and solidity, shapes and sizes are satisfied.

Figure 1.6: The inspection machine

Source: http://englishrussia.com/2013/02/06/birth-of-a-bottle/(5th March 2013)

1.2.6 Packing process

After taking out of the inspection process, glass bottles are transported into the automatic packaging machine with plastic layer sheets to prevent contamination and to ensure that they are ready to be delivered to customers in top quality, which can be seen in Figure 1.7.

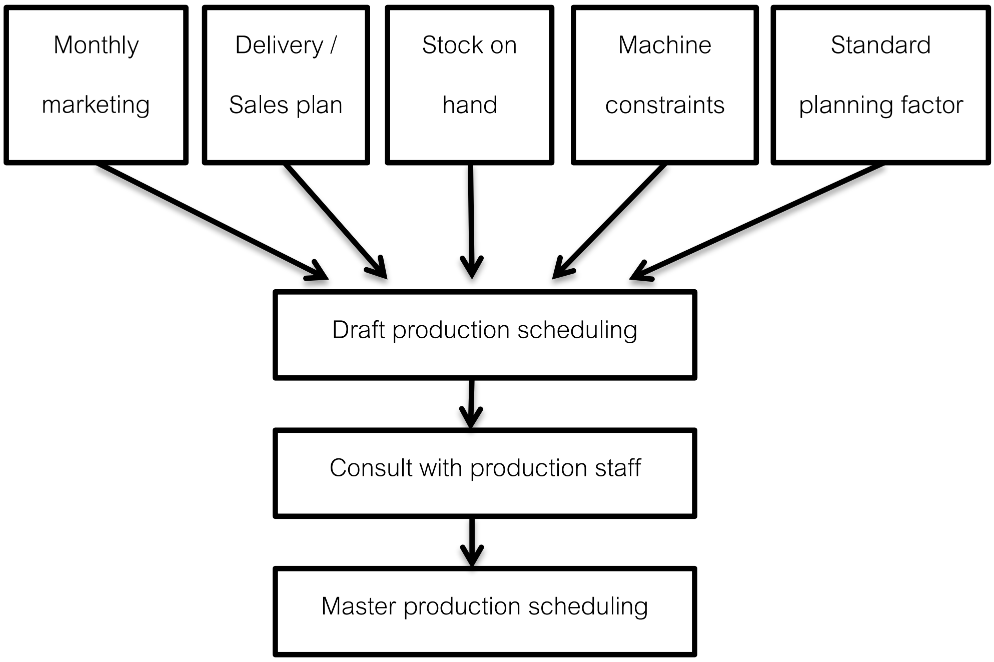
Figure 1.7: The glass bottles for shipping pallets

Source: http://en.wikipedia.org/wiki/File:Glass_bottles.jpg(5th March 2013)

1.3 Current planning strategy

Many glass manufacturers still use their production planning staff to schedule their production planning. Production planning staff receives a marketing requirements from the marketing and sales divisions. The marketing requirement consists of name of products, colours, and request quantities. The draft plan will be made based on the needs of customers, stocks on hand, machine constraints, and standard planning factors such as gob speed, gob weight, standard efficiency, and setup time. Next, the master production planning report is generated by consulting with manufacturers for ensuring the working plan that satisfied all production constraints. The current production planning work flow is shown in Figure 1.8. The solution of this strategy may not be optimized. In the case of large volumes of demand, many researchers proved that the scheduling problem with the machine-dependent setup time is a combinatorial problem or NP-hard.

Figure 1.8: The current production planning work flow



1.4 Research objective

The first objective is to formulate a mixed-integer mathematical programming model using a goal programming for minimizing the setup time and the makespan in a glass container industry. The second objective is to extend the model from Gharehgozli et *al*.[1].

1.5 Structure of the thesis

The rest of the thesis is described as follows.

In Chapter II, the theoretical backgrounds and literature review are described. This includes an important instance of the constraint relationships. The set of parameters, the decision variables and the mixed-integer mathematical programming model are presented in Chapter III. Chapter IV shown that the formulated mathematical model in Chapter III will be converted into the SAGE language. In Chapter V, the experiments and results are described and the results are discussed and analysed and the conclusion from the study are drawn in Chapter VI.

CHAPTER II

THEORETICAL BACKGROUND AND LITERATURE REVIEW

2.1. Mathematical model

We begin our discussion by giving basic definitions and notations of each part in the following linear programming problem.

$$\begin{aligned} \text{Minimize} \quad & z = c_1x_1 + c_2x_2 + \dots + c_nx_n \\ \text{subject to} \quad & a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n \geq b_1 \\ & a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n \geq b_2 \\ & \;\vdots \qquad\quad \vdots \quad + \dots + \quad \vdots \qquad \vdots \\ & a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n \geq b_m \\ & x_1, x_2, \dots, x_n \geq 0. \end{aligned}$$

The term $c_1x_1 + c_2x_2 + \dots + c_nx_n$ is the *objective function* to be minimized and will be denoted by z. The coefficients $c_1, c_2, \dots, c_n$ are the *cost coefficient* and $x_1, x_2, \dots, x_n$ are the *decision variables*. The inequality $\sum_{j=1}^{n} a_{ij}x_j \geq b_i$ denotes the *i*th *constraint*. These a_{ij} for $i = 1,2,\dots,m,\ j = 1,2,\dots,n$ form the *constraint matrix* A.

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}.$$

The constant $b_1, b_2, \dots, b_m$ are the elements in the column vector which is called the *right-hand-side* vector. The constraints $x_1, x_2, \dots, x_n \geq 0$ are the *nonnegativity constraints*. A set of variables $x_1, x_2, \dots, x_n$ satisfying all the constraints is called a *feasible point* or a *feasible solution*. The set of all feasible points constitutes the *feasible region*.

2.1.1 Decision variables

The decision variables are a set of quantities that need to be determined in order to solve the problem. Typically, the decision variables represent the amount of a resource to use. Frequently, defining decision variables of a problem is a crucial formulating step. Decision makers have some freedom to assign numerical values to decision variables because decision variables subject to constraints. Solving a mathematical model means finding these numerical values for decision variables to minimize or maximize an objective function in the presence of constraints.

2.1.2 Objective function

With the mathematical model, we wish to minimize or maximize a quantity such as cost, profit, risk, net present value, number of employees and customer satisfaction. The quantity we wish to maximize or minimize is known as an objective function. Deciding on the correct objective in practical situations is not trivial. At one extreme there may be no clear objectives. At the other, there may be clear objectives or multiple objectives.

2.1.3 Constraints

Constraints represent the limitations such as available capacities, daily working hours, raw material availability, etc. Sometimes constraints are also used to represent relationships between decision variables.

2.2 Integer programming

In this section, we will discuss an integer-programming formulation. We consider basic approaches which are binary variables and important techniques that have been developed for solving pure integer and mixed-integer programming problems. We consider an optimization problem:

Maximize $$z = \sum_{j=1}^{n} c_j x_j$$

subject to: $$\sum_{j=1}^{n} a_{ij} x_j = b_i \text{ for } i = 1,2,\ldots,m$$

$$x_j \geq 0 \text{ for } j = 1,2,\ldots,n$$

$$x_j \text{ integer for some or all } j = 1,2,\ldots,n.$$

This problem is called an *integer-programming problem*. It can be called a *mixed integer programing* when for some, but not for all, decision variables are integer variable. It is called a *pure integer program* when all decision variables are define to be integer variable. We consider basic approaches which are binary variables and important techniques that have been developed for solving pure integer and mixed-integer programming problems.

2.2.1 Binary variables

Consider the following activities which engage in (i) building a new plant, (ii) undertaking an advertising campaign, or (iii) developing a new product. In each case, a *yes–no* or so-called *go–no–go* decision must be identified. These choices are formulated by setting $x_j = 1$ if the activity is engaged and $x_j = 0$ otherwise. Decision variables that are defined to 0 or 1 are called *binary variables*. Binary variables are important because they are used in many model formulations.

If the manager decided that at most one of three activities can be pursued, the following constraint can be used.

$$\sum_{j=1}^{3} x_j \leq 1.$$

In this thesis, the binary variable is an important decision variable to use for formulating a mathematical model in the glass container industry. It can be used to find a job sequence which can be seen in Chapter III.

2.2.2 Logical constraints

Frequently, Logical constraints are used to build the model on the decision variables. The following subsections review four most used logical relationships.

2.2.2.1 Constraint feasibility

We consider a general constraint

$$f(x_1, x_2, ..., x_n) \leq b. \qquad \text{... (a)}$$

We define a binary variable α with the following:

$$\alpha = \begin{cases} 0, \text{if the constraint is satisfied,} \\ 1, \text{otherwise,} \end{cases}$$

and create the constraint

$$f(x_1, x_2, ..., x_n) - M\alpha \leq b, \qquad \text{... (b)}$$

where M is a constant which is chosen to be big enough and the constraint (b) must be satisfy if $\alpha = 1$ that is,

$$f(x_1, x_2, ..., x_n) \leq b + M,$$

for each solution of the decision variables $x_1, x_2, ..., x_n$. While the constraint (a) is satisfied when $\alpha = 0$ which gives a feasible solution to constraint (b).

2.2.2.2 Alternative constraints

Consider the constraints:

$$f_1(x_1, x_2, ..., x_n) \leq b_1,$$

$$f_2(x_1, x_2, ..., x_n) \leq b_2.$$

If at least one, but not all, of these constraints must be satisfied. We can formulate a model with this restriction by using the technique from 2.2.2.1:

$$f_1(x_1, x_2, ..., x_n) - M_1\alpha_1 \leq b_1,$$

$$f_2(x_1, x_2, ..., x_n) - M_2\alpha_2 \leq b_2,$$

$$\alpha_1 + \alpha_2 \leq 1,$$

α_1, α_2 are 0 or 1.

The decision variables α_1 and α_2 and the large constants M_1 and M_2 are used to solve when these two constraints are satisfied. The constraint $\alpha_1 + \alpha_2 \leq 1$ implies that α_1 or α_2 are equal to 0, so that at least one constraint must be satisfied.

We can eliminate one binary variable by using the relation $\alpha_1 + \alpha_2 = 1$, or equivalently, $\alpha_2 = 1 - \alpha_1$, because this formulation also implies that either α_1 or α_2 equals 0. The resulting formulation is written by:

$$f_1(x_1, x_2, ..., x_n) - M_1\alpha_1 \leq b_1,$$

$$f_2(x_1, x_2, ..., x_n) - M_2(1 - \alpha_1) \leq b_2,$$

$$\alpha_1 = 0 \text{ or } 1.$$

2.2.2.3 Conditional constraints

A conditional constraint is in the form:

$$\text{if } f_1(x_1, x_2, ..., x_n) > b_1 \text{ then } f_2(x_1, x_2, ..., x_n) \leq b_2.$$

When both $f_1(x_1, x_2, ..., x_n) > b_1$ and $f_2(x_1, x_2, ..., x_n) > b_2$, this statement is not satisfied. The conditional constraint is logically equivalent to the alternative constraints

$$f_1(x_1, x_2, ..., x_n) \leq b_1 \text{ or } f_2(x_1, x_2, ..., x_n) \leq b_2,$$

where at least one constraint must be satisfied. Therefore, this problem can be formulated by the alternative constraints as discussed in section 2.2.2.2.

2.2.2.4 *k*-fold alternative constraints

In k-fold alternative constraints technique, we want to satisfy at least *k* of the following constraints:

$$f_i(x_1, x_2, ..., x_n) \leq b_i \text{ for } i = 1, 2, ..., m.$$

For instance, these conditions may satisfy to the cold zone constraints for m inspection machines in the quality control process. If a planner has decided to use at least *k* inspection machines, then the *k* constraints for these systems must be satisfied and the remaining constraints can be ignored. Assuming that M_i for $i = 1, 2, ..., m$, are chosen. The ignored constraints can not be equal to the right-hand side values. This problem can be formulated as follows:

$$f_i(x_1, x_2, ..., x_n) - M_i(1-\alpha_i) \leq b_i \text{ for } i = 1, 2, ..., m,$$

$$\sum_{i=1}^{m} \alpha_i \geq k,$$

$$\alpha_i = 0 \text{ or } 1 \text{ for } i = 1, 2, ..., m.$$

Therefore, $\alpha_i = 1$ implies that the *i*th constraint is satisfied, and at least *k* of the constraints must be satisfied.

2.3 Goal programing

Goal programming is a branch of the multi-objective optimization. Goal programming models are very similar to linear programming models but having several objectives. It can be thought of as an extension or generalisation of linear programming to handle multiple, normally conflicting objective measures. Goal programming is used to perform three types of analysis:

1. Determining the required resources to achieve a desired set of objectives.
2. Determining the degree of attainment of the goals with the available resources.
3. Providing the best satisfying solution under a varying amount of resources and priorities of the goals.

2.4 Literature review

As discussed in [3], the single forming machine production scheduling problem calculated with the sequence-dependent setup times is experimented to be combinatorial problem. Another the sequence-dependent setup time and parallel machine production scheduling with the non-zero release time was proposed by [4]. For the case of two identical or unrelated parallel machines, it is proved that the problem of the minimal makespan is NP-hard [5] and the production scheduling problem of the minimal makespan with *m* identical or unrelated parallel machines and sequence-dependent setup times where *m* is greater than two is also NP-hard [6, 7, 8]. The phenomenon of sequence-dependent setup times has been investigated by researchers for real-World job shop environments such as the glass industry, metallurgical industry, paper industry, textile industry, wood industry and aerospace industry [9]. Several authors investigated the impact of the setup time variation of the problem [10, 11, 12].

Gharehgozli et al. (2009) [1] presented a new mixed-integer goal programming model to minimize the total weighted flow time and the total weighted tardiness simultaneously for a parallel machine scheduling problem with sequence-dependent setup times and release dates. Cheol Min Joo and Byung Soo Kim [2] presented the new mixed-integer programming model which extended from [1] and determined the allocation policy of jobs and the scheduling policy of machines to minimize the weighted sum of setup times, delay times and tardy times.

Motivated by the literatures discussed above, this thesis presents a parallel machine scheduling problem in the glass container industry with setup times and makespan by using the mixed-integer programming model and a goal programming. We extend the mathematical model from [1] for covering all of constraints in our case study.

CHAPTER III

MIXED-INTEGER PROGRAMMING MODEL FOR GLASS CONTAINER PRODUCTION SCHEDULING

This section proposes a mixed-integer mathematical programming model using a goal programming which can be used to minimize total weighted setup time and total weighted makespan of day by day. Orders are posted and requested monthly referred as jobs. We schedule all orders monthly.

3.1 Notations

Parameters

P	Set of jobs	P_k^0	Initial processing time at the machine k
MC	Set of machines	M	Big number
S_{ij}	Setup time of the job j processed directly after the job i	P_{ik}	Processing time of the job i at the machine k
S_{ik}^0	Setup time of the job i if job is the first job sequence at the machine k	w_{ik}^s, w_{ik}^m	Weights of the job i considered by setup time and makespan at the machine k
D_k	The set of jobs that cannot be assigned at the machine k		

Decision variables

ES_{ik}	The starting date of the job i at the machine k	ED_k	Dummy starting date at the machine k

$$X_{ik}^{0} = \begin{cases} 1, \text{ if the job } i \text{ processed at the first job at the machine } k \\ 0, \text{otherwise} \end{cases}$$

$$X_{ijk} = \begin{cases} 1, \text{ if the job } j \text{ process after the job } i \text{ at the machine } k \\ 0, \text{otherwise} \end{cases}$$

$$ON_{ik} = \begin{cases} 1, \text{ if the job } i \text{ is assigned at the machine } k \\ 0, \text{otherwise} \end{cases}$$

3.2 Mathematical model

The mathematical model for the glass production scheduling is

minimize $$z_1 = \sum_{i \in P} \sum_{k \in MC} w_{ik}^{s} (S_{ik}^{0} X_{ik}^{0} + \sum_{\substack{j \in P \\ j \neq i}} S_{ji} X_{jik}) \text{ and}$$

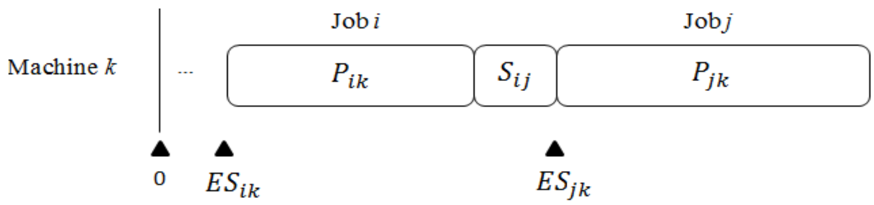
$$z_2 = \sum_{i \in P} \sum_{k \in MC} w_{ik}^{m} ED_k .$$

The objective functions of this model are classified into two goals. The objective function z_1 is used to minimize total weighted setup time. The objective function z_2 is used to minimize total weighted makespan, processing time for each job, see Figure 3.3.

These objective functions need not be minimize at the same time. In our thesis, the objective function z_1 must be included in all schedules since the setup time is the main cost of the glass container production process. The objective function z_2 is an option of production planning staff for scheduling when the demand is larger than the production rate. Next, the set of constraints is

$$ES_{ik} + P_{ik} + X_{ijk} S_{ij} \leq ES_{jk} + M(1 - X_{ijk}), \forall i, j \in P; i \neq j; \forall k \in MC \quad \text{... (1)}$$

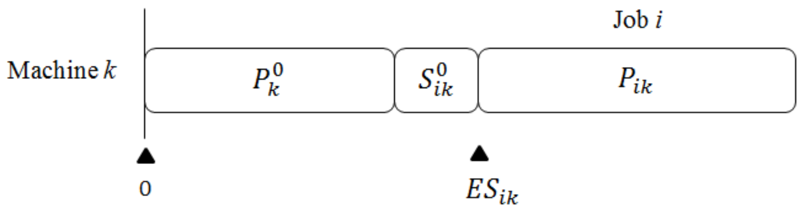
Figure 3.1: The job j starts directly after the job i on the machine k



Constraint (1) ensures the relation of jobs assigned to the same machine. The constant M is chosen to be large enough so that the constraint is satisfied if $X_{ijk} = 0$. If the job j is assigned after the job i at the machine k, $X_{ijk} = 1$, then the last term is equal to zero so this constraint is satisfied. Therefore, the sum of starting date of the job i, processing time of the job i and setup time of the job j processed after the job i is less than or equal to starting date of the job j, see Figure 3.1. The processing time is calculated from the formula as discussed in section 1.2.3. The setup time depends only on two adjacent products not their orders, i.e. $S_{ij} = S_{ji}$ for $i, j \in P$. The processing time depends only on each machine's production rate. The processing time of a machine that has a low production rate takes longer time than a machine that has a high production rate.

$$P_k^0 + S_{ik}^0 X_{ik}^0 \le ES_{ik} + M(1 - X_{ik}^0), \forall i \in P; \forall k \in MC \qquad \text{... (2)}$$

Figure 3.2: The job i at the first position on the machine k



Constraint (2) ensures that starting date of the first job in job sequence is less than or equal to sum of the processing time of last job in the previous month and its setup time, see Figure 3.2.

$$\sum_{\substack{j \in P \\ j \neq i}} X_{jik} + X_{ik}^0 = ON_{ik}, \forall i \in P; \forall k \in MC \qquad \text{... (3)}$$

Constraint (3) ensures that jobs assigned to the same machine can be appeared once in their sequence. The term $\sum_{\substack{j \in P \\ j \neq i}} X_{jik}$ means that the job i is processed adjacently after some job at the machine k. This constraint is divided into two cases.

Case 1: $ON_{ik} = 0$;

Since X_{jik} and X_{ik}^{0} are binary variables. Then $\sum_{\substack{j\in P\\ j\neq i}} X_{jik}$ and X_{ik}^{0} equal zero. This implies that the job i is not allocated on the machine k. Therefore

$$\sum_{\substack{j\in P\\ j\neq i}} X_{jik} + X_{ik}^{0} = 0+0=0=ON_{ik}.$$

Case 2: $ON_{ik} = 1$;

Case 2.1: $\sum_{\substack{j\in P\\ j\neq i}} X_{jik} = 1, X_{ik}^{0} = 0$;

This implies that the job i is processed at the machine k but it is not the first position in the job sequence

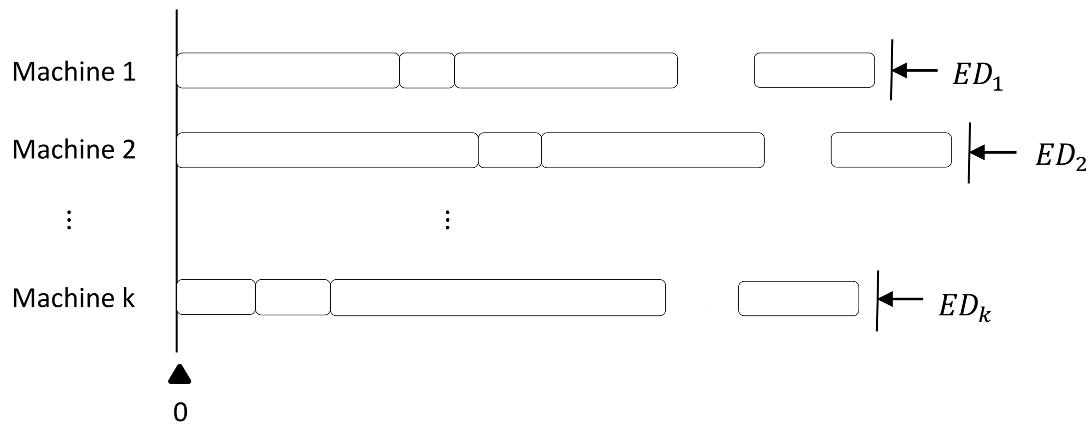
Case 2.2: $\sum_{\substack{j\in P\\ j\neq i}} X_{jik} = 0, X_{ik}^{0} = 1$;

Thus the job i is the first job in the job sequence.

$$ES_{jk} + P_{jk} \leq ED_{k} + M(1-ON_{jk}), \forall j \in P; \forall k \in MC \quad \text{... (4)}$$

$$P_{k}^{0} + S_{ik}^{0} X_{ik}^{0} + P_{ik} \leq ED_{k} + M(1-X_{ik}^{0}), \forall i \in P; \forall k \in MC \quad \text{... (5)}$$

Figure 3.3: Minimal makespan for each job sequence



Constraints (4) to (5) ensure that a dummy starting time at the machine k is greater than or equal to each starting time in a job sequence at the machine k. If the minimal makespan is required then the dummy starting time will reduce the length of each processing time so the maximum job sequence is the makespan, see Figure 3.3.

$$\sum_{k \in MC} ON_{ik} = 1, \forall i \in P \qquad \text{... (6)}$$

$$\sum_{i \in P} X^{0}_{ik} = 1, \forall k \in MC \qquad \text{... (7)}$$

Constraint (6) guarantees that each job is processed exactly on one machine. If one order generates into several jobs then number of job changes will increase accordingly. Constraint (7) ensures that last jobs in the previous month can be assigned only one job per machine.

$$\sum_{\substack{j \in P \\ j \neq i}} X_{ijk} \leq ON_{ik}, \forall i \in P; \forall k \in MC \qquad \text{... (8)}$$

Constraint (8) portrays that a machine can be derived from (7).

$$ON_{ik} = 0, \forall i \in D_k; \forall k \in MC \qquad \text{... (9)}$$

$$ES_{ik}, ED_k \geq 0, \forall i \in P; \forall k \in MC \qquad \text{... (10)}$$

$$ON_{ik}, X^{0}_{ik}, X_{ijk} = 0 \text{ or } 1, \forall i, j \in P; \forall k \in MC \qquad \text{... (11)}$$

Constraint (9) ensures the jobs that cannot be assigned at the machine k. One job can be appeared in one process type and some machine cannot be used to produce all production processes. Constraint (10) guarantees that the starting times and the dummy starting time are greater than or equal to zero. Constraint (11) ensures that ON_{ik}, X^{0}_{ik} and X_{ijk} are binary variables.

Example 3.1: Assume that there exists 4 jobs and $w^{s}_{ik} = 1 = w^{m}_{ik}$ for $i = 1,2,3,4, k = 1,2$. The parameters are shown in Tables 3.1 and 3.2. Suppose D_k is an empty set for $k = 1, 2$ and each objective function is considered separately. All processing times can be calculated from the division of requirements and production rates.

Table 3.1: Example of setup times and marketing requirements

	Setup times (days)				Requirements (tons)
	Job 1	Job 2	Job 3	Job 4	
Job 1	0	0.05	0.1	0.25	200
Job 2	0.05	0	0.75	0.05	250
Job 3	0.1	0.75	0	0.25	300
Job 4	0.25	0.05	0.25	0	350

Table 3.2: Example of Setup times with the last jobs in the previous month, production rates and processing times of the last jobs in the previous month

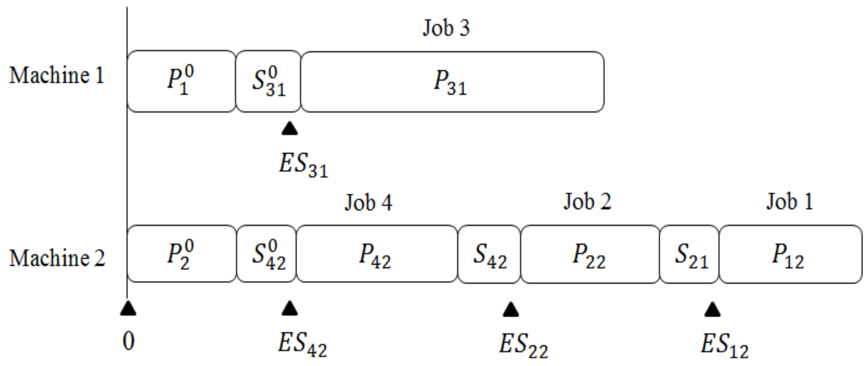
Forming machines	Setup times with jobs in the previous month (days)				Production rates (tons/day)	Processing times of last jobs in the previous month (days)
	Job 1	Job 2	Job 3	Job 4		
1	0.25	0.25	0.25	0.25	25	3
2	0.15	0.15	0.10	0.10	50	5

Table 3.3: Results after solved Example 3.1

Minimization the objective function	Forming machines	Job sequence	Starting times (days)	Processing times (days)	Setup times (days)
z_1	1	3	ES_{31} = 3.25	P_{31} = 12	S^0_{31} = 0.25
	2	4-2-1	ES_{42} = 5.10	P_{42} = 7	S^0_{42} = 0.10
			ES_{22} = 12.15	P_{22} = 5	S_{42} = 0.05
			ES_{12} = 17.20	P_{12} = 4	S_{21} = 0.05
	Total			28	0.45
z_2	1	1	ES_{11} = 3.25	P_{11} = 8	S^0_{11} = 0.25
	2	3-4-2	ES_{32} = 5.10	P_{32} = 5	S^0_{32} = 0.10
			ES_{42} = 11.35	P_{42} = 6	S_{34} = 0.25
			ES_{22} = 18.40	P_{22} = 7	S_{42} = 0.05
	Total			26	0.65

As shown in Table 3.3, the minimization of z_1 has a minimum total weighted setup time and the optimal value is 0.45 days. The minimization of z_2 has a lowest sum of total weighted processing time and total weighted setup time, i.e. it has a minimal makespan and the optimal value is 26.65 days. The results indicate that if the setup time minimization is considered then the total processing time may not be the shortest. While the makespan minimization has the lowest sum of the total weighted processing time and the total weighted setup time but total weighted setup time is not the minimum value. Example of the Gantt chart created from the optimal solution from minimization of z_1 is shown in Figure 3.4.

Figure 3.4: The Gantt chart calculated from the solution of z_1 minimization in Example 3.1



The minimal makespan is used to extend the model from [1, 2]. The last jobs in the previous month are combined into our mixed-integer mathematical programming model. Normally, it is hardly impossible that each job sequence will be completed at the same day. The first setup time effects with the setup time of the production scheduling. The processing time in [2] did not satisfy the glass container production process because the production rate of each machine is not the same. This thesis adds the machine index into the processing time parameter for satisfying with the real glass container production process. This model has $|P||MC|(|P|+4)+|P|+|MC|+\sum_{k\in MC}|D_k|$ constraints, $|P||MC|(|P|+1)$ binary variables and $|MC|(|P|+1)$ continuous variables.

CHAPTER IV

SAGE MATHEMATICAL MODEL FORMULATION

This chapter presents the conversion of the mathematical model presented in Chapter III into Python language using SAGE, a free open-source mathematic software system licensed under the GPL. It combines the power of many existing open-source packages into a common Python-based interface. SAGE covers many aspects of mathematics, including algebra, combinatorics, numerical mathematics, number theory, and calculus.

This thesis solves the glass production scheduling problem through SAGE [13] using its mixed-integer programming solver; GLPK [14]. The GLPK (GNU Linear Programming Kit) package is intended for solving large-scale linear programming, mixed-integer programming, and other related problems. It is a set of routines written in ANSI C and organized in the form of a callable library.

4.1 `MixedIntegerLinearProgram` class

As discussed in Chapter II, a mixed-integer program consists of decision variables, constraints associated with these decision variables, and an objective function which is to be maximized or minimized. The `MixedIntegerLinearProgram` class in SAGE is used for solving a mixed-integer programming problem either maximized or minimized. Four solvers are available through this class which consists of GLPK, Coin-OR, CPLEX and GUROBI. This thesis is a minimization problem and solved through GLPK solver. Define the object of our problem as `p`, the first statement in SAGE is

```
p = MixedIntegerLinearProgram(maximization=False, solver=”GLPK”).
```

… (12)

When `maximization` is set to be `True`, the `MixedIntegerLinearProgram` is defined as maximization. When `maximization` is set to be `False`, the objective function is defined as minimization.

4.2 Parameter conversion

This thesis contains the set of parameters such as jobs, forming machines, setup times, initial setup times, initial processing time, processing times, big number, and a set of jobs that cannot be assigned in some forming machines. We define

`Product`	≡	P	List of jobs
`Machine`	≡	MC	List of forming machines
`SetupTime`	≡	S_{ij}	List of setup times where the first index is for the starting job and the second index is for the target job
`StartSetupTime`	≡	S_{ik}^{0}	List of the first setup times where the first index is for the starting job and the second index is the forming machine
`Duration`	≡	P_{ik}	List of processing times where the first index is for the job and the second index is for the forming machine
`PreDuration`	≡	P_{k}^{0}	List of initial processing time
`BigM`	≡	M	Big number
`Dk`	≡	D_k	List of jobs that cannot be assigned at some forming machine where the first index is for the job and the second index is for the forming machine
`Ws,Wm`	≡	w_{ik}^{s}, w_{ik}^{m}	List of Weights of jobs considered by setup time and makespan

From Example 3.1, we can convert the set of parameters into SAGE so

```
Product = [0, 1, 2, 3]
```
… (13)

```
Machine = [0, 1]
```
… (14)

```
SetupTime = [[0, 0.05, 0.1, 0.25], [0.05, 0, 0.75, 0.05], [0.1, 0.75, 0, 0.25], [0.25, 0.05, 0.25, 0]]
```
… (15)

```
PreSetupTime = [[0.25, 0.15], [0.25, 0.15], [0.25, 0.1], [0.25, 0.1]]
```
… (16)

```
Duration = [[8, 4], [10, 5], [12, 6], [14, 7]]
```
… (17)

```
PreDuration = [3, 5]
```
… (18)

```
BigM = 1000
```
… (19)

```
Ws = [[1, 1, 1, 1], [1, 1, 1, 1]]
```
… (20)

```
Wm = [[1, 1, 1, 1], [1, 1, 1, 1]]
```
… (21)

4.3 Decision variable declaration

The statement `new_variables()` is used to declare the decision variables. We can define the type of a decision variable, binary or integer, by adding the type into its argument and also define the number of dimensions. For example, if we want to create a binary variable, namely X, with two dimensions then we can code

```
X = p.new_variables(binary=True, dim = 2).
```

If we do not input the type of the decision variable then the default type is set to continuous variable.

From Example 3.1, we define

`EarlyStart` ≡ ES_{ik} Starting time where the first index is for the job and the second index is for the forming machine

`DummyStart` ≡ ED_k Dummy starting time

`AdjFlag` ≡ $X_{ijk} = \begin{cases} 1, \text{ if the job } j \text{ process after the job } i \text{ at the machine } k \\ 0, \text{otherwise} \end{cases}$

StartAdjFlag ≡ $X_{ik}^{0} = \begin{cases} 1, \text{ if the job } i \text{ processed at the first job at the machine } k \\ 0, \text{otherwise} \end{cases}$

On ≡ $ON_{ik} = \begin{cases} 1, \text{ if the job } i \text{ is assigned at the machine } k \\ 0, \text{otherwise} \end{cases}$

We can declare the decision variables using the statement `new_variables()` so

```
EarlyStart = p.new_variables(dim=2)                                  … (22)

DummyStart = p.new_variables(dim=2)                                  … (23)

AdjFlag = p.new_variables(binary=True, dim=3)                        … (24)

StartAdjFlag = p.new_variables(binary=True, dim=2)                   … (25)

On = p.new_variables(binary=True, dim=2).                            … (26)
```

4.4 Objective function conversion

This thesis uses a goal programming with two objective functions but SAGE cannot use multi-objective function directly. We can apply a goal programming by weighting two objective functions and using a linear combination for combining into one objective function. Thus we can convert two objective functions in Chapter II,

$$z_1 = \sum_{i \in P} \sum_{k \in MC} w_{ik}^{s} (S_{ik}^{0} X_{ik}^{0} + \sum_{\substack{j \in P \\ j \neq i}} S_{ji} X_{jik}) \text{ and}$$

$$z_2 = \sum_{i \in P} \sum_{k \in MC} w_{ik}^{m} ED_k ,$$

into SAGE by using the statement `set_objective()` as follows

```
w1 = 1; w2 = 0                                                       … (27)

p.set_objective(w1*sum([PreSetupTime[i][k]*StartAdjFlag[i][k]+sum([Se
tupTime[j][i]*AdjFlag[j][i][k] for j in Product if j <> i]) for i in
Product for k in Machine]) + w2*sum([DummyStart[k][0] for k in
Machine])).                                                          … (28)
```

The `w1` and `w2` are the weights for the objective functions z_1 and z_2, respectively. For example, if we consider the objective z_1 only then we set `w1` is equal to one and set `w2` is equal to zero.

4.5 Constraint conversion

The statement `add_constraint()` is used to create the constraints. As discussed in Chapter II, we consider constraint (1) to (11) presented in Chapter III which can be converted into SAGE as follow:

$$ES_{ik} + P_{ik} + S_{ij} X_{ijk} \le ES_{jk} + M(1 - X_{ijk}), \forall i, j \in P; i \ne j; \forall k \in MC$$

```
_=[p.add_constraint(EarlyStart[i][k] + Duration[i][k] +
SetupTime[i][k]*AdjFlag[i][j][k] <= EarlyStart[j][k] + BigM*(1 -
AdjFlag[i][j][k])) for i in Product for j in Product for k in Machine
if i <> j]
```

… (29)

$$ES_{ik} + M(1 - X^0_{ik}) \ge P^0_k + S^0_{ik} X^0_{ik}, \forall i \in P; \forall k \in MC$$

```
_=[p.add_constraint(EarlyStart[i][k] + BigM*(1 - StartAdjFlag[i][k])
>= PreSetupTime[k] + PreSetupTime[i][k]*StartAdjFlag[i][k]) for i in
Product for k in Machine]
```

… (30)

$$ES_{jk} + P_{jk} \le ED_k + M(1 - ON_{jk}), \forall j \in P; \forall k \in MC$$

```
_=[p.add_constraint(EarlyStart[j][k] + Duration[j][k] <=
DummyStart[k][0] + BigM*(1 - On[j][k])) for j in Product for k in
Machine]
```

… (31)

$$P_k^0 + S_{ik}^0 X_{ik}^0 + P_{ik} \leq ED_k + M(1 - X_{ik}^0), \forall i \in P; \forall k \in MC$$

```
_=[p.add_constraint(PreDuration[k] +
PreSetupTime[i][k]*StartAdjFlag[i][k] + Duration[i][k] <=
DummyStart[k][0] + BigM*(1 - StartAdjFlag[i][k])) for i in Product
for k in Machine]
```
… (32)

$$\sum_{k \in MC} ON_{ik} = 1, \forall i \in P$$

```
_=[p.add_constraint(sum([On[i][k] for k in Machine]) == 1) for i in
Product]
```
… (33)

$$\sum_{i \in P} X_{ik}^0 = 1, \forall k \in MC$$

```
_=[p.add_constraint(sum([StartAdjFlag[i][k] for i in product]) == 1)
for k in Machine]
```
… (34)

$$\sum_{\substack{j \in P \\ j \neq i}} X_{jik} + X_{ik}^0 = ON_{ik}, \forall i \in P; \forall k \in MC$$

```
_=[p.add_constraint(sum([AdjFlag[j][i][k] for j in Product if j <>
i]) + StartAdjFlag[i][k] == On[i][k]) for i in Product for k in
Machine]
```
… (35)

$$\sum_{\substack{j \in P \\ j \neq i}} X_{ijk} \leq ON_{ik}, \forall i \in P; \forall k \in MC$$

```
_=[p.add_constraint(sum([AdjFlag[i][j][k] for j in Product if j <>
i]) <= On[i][k]) for i in Product for k in Machine]
```
… (36)

$$ON_{ik} = 0, \forall i \in D_k ; \forall k \in MC$$

```
for k in Machine:
    _=[p.add_constraint(On[i][k] == 0) for i in D[k]]
```
… (37)

Constraint (10) and (11) are not input in the constraint part because each decision variable have assigned to positive values in the step of decision variable declaration. When the model components are completed the equations and inequalities from (12) to (37) can be solved by the statement

```
p.solve().
```

All the objective function and the constraints can be showed by typing he command

```
p.show().
```

CHAPTER V

EXPERIMENTS AND COMPUTATIONAL RESULTS

To evaluate the performances of the mixed-integer mathematical programming model proposed in this thesis, the previous production planning from three months of the company in Thailand were used to compare, data set I to III. These parameters can be seen in Table 4.1.

Table 4.1: Details of data set I to III

Data sets	Jobs	Forming machines	Number of constraints	Number of binary variables	Number of continuous variables
I	11	4	679	528	48
II	12	8	1,660	1,248	104
III	15	8	2,311	1,920	128

SAGE version 5.4.1 was used for finding the optimal solution. These tests have been done on a portable computer with Intel Core i7 with 8GB RAM and MS Windows 8 Operating System. Experiments proved that average the average setup time loss calculated from the mixed-integer programming model presented in Chapter III is better than the actual production scheduling, see Table 4.2.

The comparison considers the objective function z_1 only (minimize setup time). Minimal makespan optimization was not used in our case study because production rate and demand are not different but the makespan calculated from our case study is shown in Table 4.3.

Table 4.2: Compare the total setup time with the actual production scheduling for three months

Data sets	Total setup time (minutes)		Gain = Actual - Model (minutes)	%Gain
	Actual	Model		
I	1,713.6	1,598.4	115.2	7.21%
II	1,972.8	1,843.2	129.6	7.03%
III	2,476.8	2,304.0	172.8	7.50%
Grand total	6,163.2	5,745.6	417.6	7.27%
Average	2,054.4	1,915.2	139.2	7.27%

Table 4.3: Makespan calculated from data set I to III

Data sets	$\sum_{k \in MC} ED_k$ (days)	$Gain = \sum_{k \in MC} (ED_k - 30)$ (days)
I	105.15	-14.85
II	159.32	-82.01
III	N/A	-

Table 4.3 implies that the production planning staff can assign the other orders to the machine available times. For example, the data set I can be assigned the other orders to fulfil the machine available times, -14.85 days. The N/A in Table 4.3 means the total weighted makespan of the data set III cannot be calculated with our portable computer because the internal memories are not enough to calculate the solution.

The linear combination of objective functions, $u_1 z_1 + u_2 z_2$, are considered for analysing sensitivity when the production planning staff want to use both objective functions. For example, the value of u_1 and u_2 are varied from 0 to 1 and used for the data set I, see Table 4.4.

Table 4.4: Compare the total setup time and the makespan of data set I by varying weights of objective functions

u_1	u_2	Total setup time (days)	Makespan (days)
1	0	1.11	114.11
0.95	0.05	1.15	105.15
0.75	0.25	1.15	105.15
0.50	0.50	1.15	105.15
0.25	0.75	1.15	105.15
0	1	1.15	105.15

The first row of Table 4.4, $u_1 = 1$ and $u_2 = 0$, means that the model minimizes the objective function z_1 only. The result after solved this case show that the makespan is not calculated because the dummy starting date was not combine in the related jobs constraint which can be seen in the constraint (1) and (2) presented in Chapter III. The total setup time in this case has the minimum value, 1.11 days, but the makespan may not be minimized. The case $u_1 > 0$ and $u_2 > 0$ and the case $u_1 = 0$ and $u_2 = 1$ have the same optimal solution because the setup time is implicated in the makespan constraints, constraints (4) and (5) presented in Chapter III, which causes the setup time and the makespan to be minimized at the same time. The case $u_1 = 1$ and $u_2 = 0$ and the case $u_1 = 0$ and $u_2 = 1$ may have the same optimal solution where each machine has high production rate and gob weight and process type of each job are similar.

CHAPTER VI

CONCLUSION

In this thesis, the parallel machine scheduling with setup time and makespan in the glass container industry is considered. The first objective is to formulate a mixed-integer programming model using a goal programming for minimizing totoal weighted setup time and total weighted makespan. The second objective is to extend the model from Gharehgozli et al. [1] for covering all of constraints in the glass container production process.

The test results indicate that the average setup time calculated from the mixed-integer mathematical programming model is better than the actual production scheduling from the company in Thailand 7.27%. If the production planning department still uses staffs to construct the production scheduling then the optimal solution may not be obtained because the production scheduling with machine-dependent setup time is a combinatorial problem. Therefore, the mixed-integer mathematical programming model can be used to help the production planning department for finding the optimal solution in the case of a high volume of orders. This thesis adds the minimal makespan condition for using in the case of the demand exceeds the production rate. The experiment proved that the minimization makespan can be used for reducing the processing times for each the job sequences. The planning staff can use two objective functions separately by setting the weight of another objective to be zero.

REFERENCES

[1] Gharehgozli, A.H., Tavakkoli-Moghaddam, R. and Zaerpour, N. A fuzzy-mixed-integer goalprogramming mode for a parallel-machine scheduling problem with sequence-dependent setup times and release dates. **Robotics and computer-Integrated Manufacturing**, No.25 (2009): 853-859.

[2] Cheol Min Joo and Byung Soo Kim. Parallel machine scheduling problem with ready time, due times and sequence-dependent setup times using meta-heuristic algorithms. **Engineering Optimization**, No.11 (2011): 1-14.

[3] Pinado, M. Scheduling theory, algorithms, and systems. **Upper Saddle River, NJ: Prentice- Hall**, 1995.

[4] Nessah, F., Yalaoui, F. and Chu, C. New heuristics for identical parallel machine scheduling with sequencedependent setup times and dates. **Proceedings of the international conference on industrial engineering and systems. management** (2005): 16–19.

[5] Garey, M. and Johnson, D. Computers and intractability: A guide to the theory of NP-completeness. **New York: W.H. Freeman**, 1997.

[6] Nait, T.D., Chu, C.,Yalaoui, F. and Amodeo, L. A new approach for identical parallel machine scheduling with job splitting and sequence-dependent setup times based on linear programming. **International conference on industrial engineering and production management**, No.3 (2003): 266–274.

[7] Sveltana, A., Kravchenko, S. and Werner, F. A heuristic algorithm for minimizing mean flow time with unit setups. **Information Processing Letters**, No.79 (2001): 291–296.

[8] Yalaoui, F. and Chu, C.. An efficient heuristic approach for parallel machine scheduling with job splitting and sequence-dependent setup times. **IIE Transactions**, 35 (2) (2003): 183–190.

[9] Behnamian, J., Zandieh M. and Ghomi, F. Parallel-machine scheduling problems with sequence-dependent setup times using an ACO, SA and VNS hybrid algorithm. **Expert Systems with Applications**, No.36 (2009): 9637-9644.

[10] Bobrowski, P.M. and Kim, S. C. The impact of sequence dependent setup time on job shop scheduling performance. **International Journal of Production Research**, No.32 (1994): 1503–1520.

[11] Flynn, B.B. The effects of setup time on output capacity in cellular manufacturing. **International Journal of Production Research**, No.25 (1987): 1761–1772.

[12] Hahn, C.K., Bragg, D.J., and Shin, D.W. Impact of the setup variable on capacity and inventory decisions. **Academic Management Review**, No.13 (1989): 91–103.

[13] Stein, W. Sage Mathematics Software (Version 5.4.1). **The Sage Group.** (2013): http://www.sagemath.org

[14] Makhorin, A. GLPK-GNU Linear Programming Kit. **The GNU Group.** (2008): http://www.gnu.org/software/glpk/.

APPENDICES

Appendix A: Setup times of data set I to III (days)

Data set I

S_{ij}	1	2	3	4	5	6	7	8	9	10	11
1	0.00	0.09	0.09	0.09	0.12	0.10	0.10	0.10	0.09	0.09	0.09
2	0.10	0.00	0.10	0.10	0.14	0.13	0.13	0.13	0.10	0.10	0.10
3	0.10	0.10	0.00	0.10	0.14	0.13	0.13	0.13	0.10	0.10	0.10
4	0.10	0.10	0.10	0.00	0.14	0.13	0.13	0.13	0.10	0.10	0.10
5	0.13	0.13	0.13	0.13	0.00	0.13	0.13	0.13	0.13	0.13	0.13
6	0.19	0.19	0.19	0.19	0.5	0.00	0.10	0.10	0.19	0.19	0.19
7	0.19	0.19	0.19	0.19	0.5	0.10	0.00	0.10	0.19	0.19	0.19
8	0.19	0.19	0.19	0.19	0.5	0.10	0.10	0.00	0.14	0.13	0.13
9	0.10	0.10	0.10	0.10	0.14	0.13	0.13	0.13	0.00	0.10	0.10
10	0.10	0.10	0.10	0.10	0.14	0.13	0.13	0.13	0.10	0.00	0.10
11	0.10	0.10	0.10	0.10	0.14	0.13	0.13	0.13	0.10	0.10	0.00

<u>Data set II</u>

S_{ij}	1	2	3	4	5	6	7	8	9	10	11	12
1	0.00	0.10	0.19	0.19	0.19	0.19	0.10	0.10	0.10	0.10	0.19	0.19
2	0.10	0.00	0.19	0.19	0.19	0.19	0.10	0.10	0.10	0.10	0.19	0.19
3	0.13	0.13	0.00	0.10	0.10	0.10	0.13	0.13	0.13	0.13	0.10	0.10
4	0.10	0.10	0.09	0.00	0.09	0.09	0.10	0.10	0.10	0.10	0.09	0.09
5	0.13	0.13	0.10	0.10	0.00	0.10	0.13	0.13	0.13	0.13	0.10	0.10
6	0.13	0.13	0.10	0.10	0.10	0.00	0.13	0.13	0.13	0.13	0.10	0.10
7	0.10	0.10	0.19	0.19	0.19	0.19	0.00	0.10	0.00	0.10	0.19	0.19
8	0.10	0.10	0.14	0.14	0.14	0.14	0.10	0.00	0.10	0.10	0.14	0.14
9	0.10	0.10	0.19	0.19	0.19	0.19	0.10	0.10	0.00	0.10	0.19	0.19
10	0.10	0.10	0.19	0.19	0.19	0.19	0.10	0.10	0.10	0.00	0.19	0.19
11	0.13	0.13	0.10	0.10	0.10	0.10	0.13	0.13	0.13	0.13	0.00	0.10
12	0.13	0.13	0.10	0.10	0.10	0.10	0.13	0.13	0.13	0.13	0.10	0.00

<u>Data set III</u>

S_{ij}	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0.00	0.19	0.10	0.10	0.19	0.19	0.19	0.19	0.10	0.10	0.10	0.10	0.19	0.19	0.50
2	0.10	0.00	0.10	0.10	0.09	0.09	0.09	0.09	0.10	0.10	0.10	0.10	0.09	0.09	0.12
3	0.10	0.19	0.00	0.10	0.19	0.19	0.19	0.19	0.10	0.10	0.10	0.10	0.19	0.19	0.50
4	0.10	0.19	0.10	0.00	0.19	0.19	0.19	0.19	0.10	0.10	0.10	0.10	0.19	0.19	0.5
5	0.13	0.10	0.13	0.13	0.00	0.10	0.10	0.10	0.13	0.13	0.13	0.13	0.10	0.10	0.14
6	0.10	0.09	0.10	0.10	0.5	0.00	0.09	0.09	0.10	0.10	0.10	0.10	0.09	0.09	0.12
7	0.13	0.10	0.13	0.13	0.5	0.10	0.00	0.10	0.13	0.13	0.13	0.13	0.10	0.10	0.14
8	0.13	0.10	0.13	0.13	0.5	0.10	0.10	0.00	0.13	0.13	0.13	0.13	0.10	0.10	0.14
9	0.10	0.19	0.10	0.10	0.19	0.19	0.19	0.19	0.00	0.10	0.10	0.10	0.19	0.19	0.5
10	0.10	0.14	0.10	0.10	0.14	0.14	0.14	0.14	0.10	0.00	0.10	0.10	0.14	0.14	0.5
11	0.10	0.19	0.10	0.10	0.19	0.19	0.19	0.19	0.10	0.10	0.00	0.10	0.19	0.19	0.5
12	0.10	0.19	0.10	0.10	0.19	0.19	0.19	0.19	0.10	0.10	0.10	0.00	0.19	0.19	0.5
13	0.13	0.10	0.13	0.13	0.10	0.10	0.10	0.10	0.13	0.13	0.13	0.13	0.00	0.10	0.14
14	0.13	0.10	0.13	0.13	0.10	0.10	0.10	0.10	0.13	0.13	0.13	0.13	0.10	0.00	0.14
15	0.13	0.13	0.13	0.13	0.5	0.13	0.13	0.13	0.13	0.13	0.13	0.13	0.13	0.13	0.00

Appendix B: Processing times (P_{ik}) of data set I to III (days)

<u>Data set I</u>

Jobs	Forming machines			
	1	2	3	4
1	10	10	7	5
2	13	13	10	7
3	12	12	9	6
4	7	7	5	3
5	4	4	4	2
6	14	14	10	7
7	8	8	6	4
8	12	12	8	6
9	14	14	11	7
10	10	10	8	5
11	21	21	16	11

Data set II

Jobs	Forming machines							
	1	2	3	4	5	6	7	8
1	6	6	6	8	8	6	4	4
2	9	9	9	12	12	9	6	6
3	8	8	8	11	11	8	5	5
4	10	10	10	13	13	10	7	7
5	4	4	4	6	6	4	3	3
6	6	6	6	8	8	6	4	4
7	6	6	6	8	8	6	4	4
8	7	7	7	9	9	7	4	4
9	5	5	5	7	7	5	3	3
10	9	9	9	12	12	9	6	6
11	14	14	14	18	18	14	9	9
12	8	8	8	11	11	8	5	5

Data set III

Jobs	Forming machines							
	1	2	3	4	5	6	7	8
1	28	28	28	44	44	28	21	21
2	19	19	19	26	26	19	13	13
3	7	7	7	9	9	7	4	4
4	11	11	11	15	15	11	7	7
5	9	9	9	12	12	9	6	6
6	11	11	11	15	15	11	7	7
7	6	6	6	7	7	6	4	4
8	7	7	7	9	9	7	5	5
9	7	7	7	9	9	7	5	5
10	8	8	8	11	11	8	5	5
11	6	6	6	8	8	6	4	4
12	10	10	10	14	14	10	7	7
13	15	15	15	20	20	15	10	10
14	9	9	9	12	12	9	6	6
15	7	7	7	7	7	7	4	4

Appendix C: Setup times with the last jobs in the previous month, production rates and processing times of the last jobs in the previous month of data set I to III

Data set I

Forming machines	S_{ik}^{0} (days)											P_{k}^{0} (days)
	Job 1	Job 2	Job 3	Job 4	Job 5	Job 6	Job 7	Job 8	Job 9	Job 10	Job 11	
1	0.09	0.10	0.10	0.10	0.13	0.19	0.19	0.19	0.10	0.10	0.10	4
2	0.09	0.10	0.10	0.10	0.13	0.19	0.19	0.19	0.10	0.10	0.10	9
3	0.10	0.13	0.13	0.13	0.13	0.10	0.10	0.10	0.13	0.13	0.13	2
4	0.09	0.10	0.10	0.10	0.13	0.19	0.19	0.19	0.10	0.10	0.10	5

Data set II

Forming machines	S_{ik}^{0} (days)												P_{k}^{0} (days)
	Job 1	Job 2	Job 3	Job 4	Job 5	Job 6	Job 7	Job 8	Job 9	Job 10	Job 11	Job 12	
1	0.10	0.10	0.19	0.19	0.19	0.19	0.10	0.10	0.10	0.10	0.19	0.19	16
2	0.10	0.10	0.09	0.09	0.09	0.09	0.10	0.10	0.10	0.10	0.09	0.09	7
3	0.10	0.10	0.19	0.19	0.19	0.19	0.10	0.10	0.10	0.10	0.19	0.19	2
4	0.13	0.13	0.10	0.10	0.10	0.10	0.13	0.13	0.13	0.13	0.10	0.10	17
5	0.13	0.13	0.10	0.10	0.10	0.10	0.13	0.13	0.13	0.13	0.10	0.10	14
6	0.10	0.10	0.19	0.19	0.19	0.19	0.10	0.10	0.10	0.10	0.19	0.19	21
7	0.10	0.10	0.19	0.19	0.19	0.19	0.10	0.10	0.10	0.10	0.19	0.19	8
8	0.13	0.13	0.10	0.10	0.10	0.10	0.13	0.13	0.13	0.13	0.10	0.10	15

Data set III

Forming machines	S_{ik}^{0} (days)												P_k^0 (days)
	Job 1	Job 2	Job 3	Job 4	Job 5	Job 6	Job 7	Job 8	Job 9	Job 10	Job 11	Job 12	
1	0.10	0.19	0.10	0.10	0.19	0.19	0.19	0.19	0.10	0.10	0.10	0.10	2
2	0.10	0.09	0.10	0.10	0.09	0.09	0.09	0.09	0.10	0.10	0.10	0.10	11
3	0.10	0.19	0.10	0.10	0.19	0.19	0.19	0.19	0.10	0.10	0.10	0.10	12
4	0.13	0.10	0.13	0.13	0.10	0.10	0.10	0.10	0.13	0.13	0.13	0.13	18
5	0.13	0.10	0.13	0.13	0.10	0.10	0.10	0.10	0.13	0.13	0.13	0.13	6
6	0.10	0.19	0.10	0.10	0.19	0.19	0.19	0.19	0.10	0.10	0.10	0.10	9
7	0.10	0.19	0.10	0.10	0.19	0.19	0.19	0.19	0.10	0.10	0.10	0.10	23
8	0.13	0.10	0.13	0.13	0.10	0.10	0.10	0.10	0.13	0.13	0.13	0.13	9

Forming machines	S_{ik}^{0} (days)		
	Job 13	Job 14	Job 15
1	0.19	0.19	0.50
2	0.09	0.09	0.12
3	0.19	0.19	0.50
4	0.10	0.10	0.14
5	0.10	0.10	0.14
6	0.19	0.19	0.50
7	0.19	0.19	0.50
8	0.10	0.10	0.14

BIOGRAPHY

Name	Chaowalit Bunchom
Date of Birth	25 June 1985
Place of Birth	Trang, Thailand
Education	B.Sc.(Mathematic), Chulalongkorn University, 2007
Scholarship	Bangkok Glass Co., Ltd.
Publication	C. Bunchom, K. Sinapiromsaran, *Mixed Integer Mathematical Programming Model for the Glass Container Production Scheduling*, in:Proc. 18^{th} Annual Meeting in Mathematics, Krabi (2013).