การจัดตารางของกระบวนการอัดแรงดันในการผลิตแผ่นวงจรพิมพ์หลายชั้นด้วยเอ็มไอแอลพีและฮิวริสติก

นายธีระเดช ไหลสุพรรณวงศ์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรดุษฎีบัณฑิต

สาขาวิชาคณิตศาสตร์ประยุกต์และวิทยาการคณนา

ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์

คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

ปีการศึกษา 2564

SCHEDULING OF PRESSING PROCESS IN MULTI-LAYER PRINTED CIRCUIT BOARD MANUFACTURING VIA MILP AND HEURISTIC

Mr. Teeradech Laisupannawong

A Dissertation Submitted in Partial Fulfillment of the Requirements

for the Degree of Doctor of Philosophy Program in Applied Mathematics and

Computational Science

Department of Mathematics and Computer Science

Faculty of Science

Chulalongkorn University

Academic Year 2021

Dissertation Title	SCHEDULING OF PRESSING PROCESS IN MULTI-LAYER PRINTED CIRCUIT BOARD MANUFACTURING VIA MILP AND HEURISTIC
By	Mr. Teeradech Laisupannawong
Field of Study	Applied Mathematics and Computational Science
Dissertation Advisor	Assistant Professor Boonyarit Intiyot, Ph.D.
Dissertation Co-advisor	Associate Professor Chawalit Jeenanunta, Ph.D.

Accepted by the Faculty of Science, Chulalongkorn University in Partial Fulfillment of the Requirements for the Doctoral Degree

Polkit Sangvanich .... Dean of the Faculty of Science

(Professor Polkit Sangvanich, Ph.D.)

DISSERTATION COMMITTEE

[illegible] .... Chair External Examiner

(Associate Professor Jirachai Buddhakulsomsiri, Ph.D.)

[illegible] .... Dissertation Advisor

(Assistant Professor Boonyarit Intiyot, Ph.D.)

[illegible] .... Dissertation Co-advisor

(Associate Professor Chawalit Jeenanunta, Ph.D.)

[illegible] .... Committee

(Associate Professor Krung Sinapiromsaran, Ph.D.)

[illegible] .... Committee

(Associate Professor Phantipa Thipwiwatpotjana, Ph.D.)

[illegible] .... Committee

(Assistant Professor Kitiporn Plaimas, Ph.D.)

ธีระเดช ไหลสุพรรณวงศ์ : การจัดตารางของกระบวนการอัดแรงดันในการผลิตแผ่นวงจรพิมพ์หลายชั้นด้วยเอ็มไอแอลพีและฮิวริสติก. (SCHEDULING OF PRESSING PROCESS IN MULTI-LAYER PRINTED CIRCUIT BOARD MANUFACTURING VIA MILP AND HEURISTIC) อ.ที่ปรึกษาวิทยานิพนธ์หลัก : ผศ.ดร.บุญฤทธิ์ อินทิยศ, อ.ที่ปรึกษาวิทยานิพนธ์ร่วม : รศ.ดร.ชวลิต จีนอนันต์ 121 หน้า.

กระบวนการอัดแรงดันมีวัตถุประสงค์ในการอัดพาเนลซึ่งเกิดจากการประกบกันของส่วนประกอบต่าง ๆ เพื่อสร้างแผ่นวงจรพิมพ์หลายชั้น กระบวนการนี้เป็นส่วนหนึ่งในการผลิตแผ่นวงจรพิมพ์หลายชั้นและสามารถจัดเป็นปัญหาการจัดตารางที่มีวัตถุประสงค์ให้ค่าเมคสแปนมีค่าน้อยที่สุด วิทยานิพนธ์นี้นำเสนอตัวแบบกำหนดการเชิงเส้นเชิงจำนวนเต็มแบบผสมสองตัวแบบ (ตัวแบบที่หนึ่งและตัวแบบที่สอง) และฮิวริสติกอัลกอลิทึมที่มีชื่อว่า three-phase-PCB-pressing heuristic (3P-PCB-PH) สำหรับการจัดตารางกระบวนการอัดแรงดัน ตัวแบบที่สองเป็นการปรับปรุงจากตัวแบบที่หนึ่งในแง่ของขนาดของตัวแบบและมิติของตัวแปรตัดสินใจบางตัว ตัวแบบทั้งสองและอัลกอลิทึม 3P-PCB-PH ถูกนำมาใช้แก้ปัญหาทดสอบที่สร้างจากข้อมูลจริงจากบริษัทผู้ผลิตแผ่นวงจรพิมพ์แห่งหนึ่ง ผลการทดลองพบว่าตัวแบบที่สองสามารถหาผลเฉลยที่เหมาะที่สุดสำหรับปัญหาทดสอบได้หลายปัญหามากกว่าตัวแบบที่หนึ่ง และมีประสิทธิภาพดีกว่าตัวแบบที่หนึ่งในแง่ความซับซ้อนเชิงขนาดและความซับซ้อนเชิงการคำนวณโดยใช้เวลาในการคำนวณเร็วกว่าตัวแบบที่หนึ่งโดยเฉลี่ย 34.71% นอกจากนี้ อัลกอลิทึม 3P-PCB-PH สามารถแก้ทุกปัญหาทดสอบและให้ผลเฉลยที่เหมาะที่สุดหรือผลเฉลยที่ใกล้เหมาะที่สุดโดยใช้เวลาในการคำนวณน้อยกว่า 1 วินาที ซึ่งเหมาะสมมากกับการนำไปใช้ในทางปฏิบัติในอุตสาหกรรมจริงของการผลิตแผ่นวงจรพิมพ์

ภาควิชา	คณิตศาสตร์และ	ลายมือชื่อนิสิต	ธีระเดช ไหลสุพรรณวงศ์
	วิทยาการคอมพิวเตอร์	ลายมือชื่อ อ.ที่ปรึกษาหลัก	[illegible]
สาขาวิชา	คณิตศาสตร์ประยุกต์	ลายมือชื่อ อ.ที่ปรึกษาร่วม	[illegible]
	และวิทยาการคณนา		
ปีการศึกษา	2564		

6172822123 : MAJOR APPLIED MATHEMATICS AND COMPUTATIONAL SCIENCE

KEYWORDS : PRESSING PROCESS / PRINTED CIRCUIT BOARD / SCHEDULING / MIXED-INTEGER LINEAR PROGRAMMING / HEURISTIC

TEERADECH LAISUPANNAWONG : SCHEDULING OF PRESSING PROCESS IN MULTI-LAYER PRINTED CIRCUIT BOARD MANUFACTURING VIA MILP AND HEURISTIC. ADVISOR : ASST. PROF. BOONYARIT INTIYOT, Ph.D., DISSERTATION COADVISOR : ASSOC. PROF. CHAWALIT JEENANUNTA, Ph.D., 121 pp.

The pressing process aims to press the panel which is the stack of materials to form a multi-layer printed circuit board (PCB). This process is a part of multi-layer PCB fabrication and can be considered as a scheduling problem with the objective of minimizing the makespan. In this dissertation, two mixed-integer linear programming models (Models 1 and 2) and a three-phase-PCB-pressing heuristic (3P-PCB-PH) algorithm for scheduling the pressing process are presented. Model 2 is an improvement of Model 1 in terms of the model size and the dimensionality of some decision variables. Both models and the 3P-PCB-PH algorithm are used to solve the test problems that are generated from the actual data from a PCB company. The results show that Model 2 can find an optimal solution in more test problems than Model 1 and outperform Model 1 in terms of the size complexity and the computational complexity with 34.71% average relative improvement of the computational time. Moreover, the 3P-PCB-PH algorithm can solve all test problems and give an optimal solution or a near optimal solution using the computational time of less than 1 second, which is very practical in the real PCB manufacturing industry.

Department	: Mathematics and Computer Science	Student's Signature	Teeradech Laisupannawong
Field of Study	: Applied Mathematics and Computational Science	Advisor's Signature	[illegible]
Academic Year	: 2021	Co-advisor's Signature	[illegible]

ACKNOWLEDGEMENTS

Firstly, I would like to thank my advisor, Assistant Professor Dr. Boonyarit Intiyot, and my co-advisor, Associate Professor Dr. Chawalit Jeenanunta, for their invaluable help, guidance, and supervision. They always gave excellent advice and encouragement when I had difficulties and problems about this research. I could not accomplish this dissertation without their expertise and support.

I also would like to thank all of my dissertation committees, Associate Professor Dr. Jirachai Buddhakulsomsiri, Associate Professor Dr. Krung Sinapiromsaran, Associate Professor Dr. Phantipa Thipwiwatpotjana, and Assistant Professor Dr. Kitiporn Plaimas for their recommendations and suggestions, which were very helpful in developing and improving this research.

In addition, I am grateful to all instructors who educated me all along. I am also thankful to my friends, AMCS students, for their suggestions and their help, especially Mr. Thanet Markchom who helped me in English and Dr. Supphakorn Sumetthapiwat who helped me in computer programming.

Moreover, I would like to give thanks to the Development and Promotion of Science and Technology Talents Project (DPST) for funding and supporting all of my Ph.D. program study.

Finally, I am most grateful to my family, especially my mother and brother for their encouragement and support. They always foster me throughout the study and everything in my life.

CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

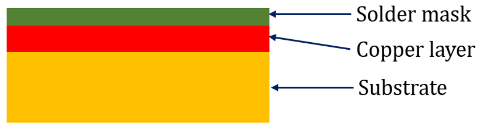
CHAPTER I

INTRODUCTION

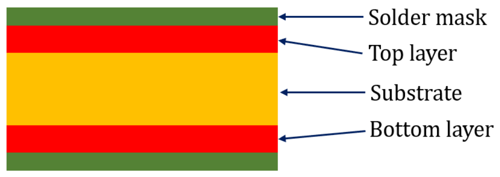
1.1 Motivation

A printed circuit board (PCB) plays an important role in modern times [1]. PCBs are boards used to connect electronic devices and are essential parts of various electronic products in everyday life, such as air conditioners, televisions, refrigerators, computers, and mobile phones. Because of the high demand of these electronic products, the demand of PCBs increases rapidly at present. It is therefore no surprise that the PCB production has become a competitive market [2]. Almost all PCB manufacturers are encountering the challenge of enhancing production efficiency to deal with the intense competition. In every aspect of production, the productiveness and cost are important points. Hence, every PCB manufacturer aims to reduce the manufacturing cost and to maximize the efficiency of production machines.

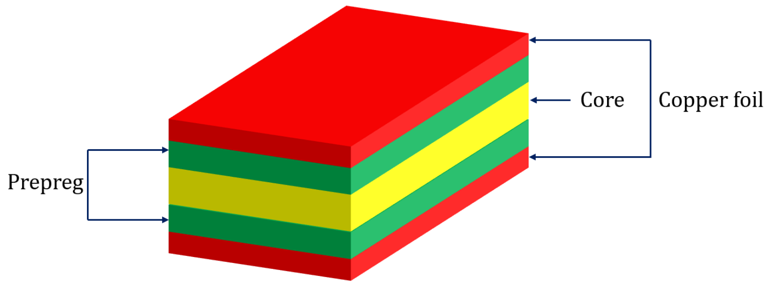
In accordance with the number of layers, PCBs can be characterized into three types as single-layer PCBs, double-layer PCBs, and multi-layer PCBs. Figure 1.1 shows each type of PCBs. Single-layer PCBs have just one layer of base material and consist of circuits and electronic components on the only single side. They are mostly used in simple circuitry, such as calculators, radios, and printers. In double-layer PCBs, the circuit pattern is available on both sides. This type of PCBs is mostly used in industrial control, converter, and LED lighting. The multi-layer PCBs expand the technology used in double-layer boards. This PCB type has more than two layers. It is very complex and used in large applications like satellite systems, GPS technology, medical equipment, data storage, and computers. Each PCB type has different manufacturing procedures and consists of many stages or processes. A PCB manufacturer must provide a good scheduling plan for any process in the manufacturing so that the production time and



(a) Single-layer PCB



(b) Double-layer PCB



(c) Multi-layer PCB

Figure 1.1: Types of PCB.

cost are reduced.

Multi-layer PCBs are the most complex type of PCBs and are used in high technology products. Their manufacturing procedure consists of many stages, and their production cost is very high because of the complexity of the manufacturing process. According to Khandpur [3], multi-layer PCB manufacturing has four stages as shown in Figure 1.2, which include the design, the fabrication, the assembly, and the testing. The design stage aims to create the circuit patterns using a PCB design software. The fabrication stage focuses on constructing a PCB or bare board. The assembly stage aims to set electronic devices, such as transistors, resistors, and capacitors on specified positions on a bare PCB. Lastly, the PCB testing is the stage of inspection to assure that PCBs meet the customer's requirements and standards. Note that each stage also contains many processes. Every PCB company tries to reduce the cost or production time in every process but still maintains the product quality.

In multi-layer PCB manufacturing, the pressing process is a part of the fabrication stage. It is a time- and cost-consuming process and consists of many materials and costly machines. In the pressing process, the production time can be reduced while the utilization of machines can be increased by providing a suitable schedule. In reality, the pressing

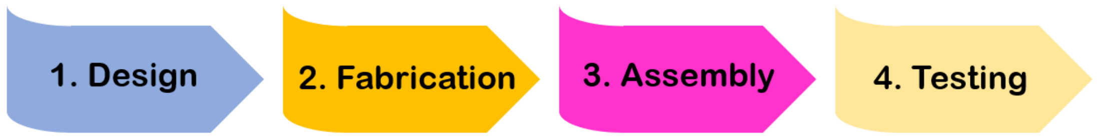


Figure 1.2: Multi-layer PCB manufacturing.

process is scheduled manually by PCB staffs, which may not yield the best utilization of resources. Mathematical optimization can help produce an optimal schedule, which uses assignment and sequencing.

Despite the importance of the pressing process scheduling in the PCB manufacturing industry, there is no research study in the literature that investigates the pressing process scheduling. Therefore, this dissertation focuses on the pressing process scheduling and aims to fulfill this gap by providing some approaches for solving the pressing process scheduling with the objective of maximizing the resource utilization. These approaches will provide an option for an optimal schedule or a good schedule for the pressing process in any PCB manufacturing industries to reduce their production time and cost.

1.2 Background Knowledge

In this section, the background knowledge for this research study is presented. The details of multi-layer PCB fabrication are described in the first section. Then, a brief background in linear programming, nonsimultaneous constraints, and heuristic are given. Next, a flexible job-shop scheduling problem, which is a type of scheduling problems and has similar backgrounds as the pressing process scheduling, is introduced. Lastly, the comparison of the performance of mixed-integer linear programming models is explained.

1.2.1 Multi-layer PCB fabrication

Multi-layer PCB fabrication is a stage in multi-layer PCB manufacturing. This stage aims to construct PCB or a bare board. A variety of processes are currently used for fabricating multi-layer PCBs by each PCB manufacturer. However, most of the processes are identical or similar. According to Khandpur [3], copper clad laminate, prepreg, and copper foil are the main materials in the multi-layer PCB fabrication. The fabrication of multi-layer PCBs is comprised of the following steps:

1. The sheets of copper clad laminate are cut to the proper size in the cutting process.
2. The original designed pattern is transferred to the copper surface of the laminate sheet, and the copper is then removed from undesirable areas in order to get the desired circuit pattern on the laminate in the etching process. In this dissertation, an etched laminate is called a *core*.
3. A number of cores are stacked together, and a prepreg is put between each pair of them. Also, a copper foil is placed on the top and bottom layers. In this dissertation, this stack is called a *panel*. The panel is pressed using pressure and heat in the pressing process to form a multi-layer board. More details about the pressing process will be described in Chapter 3.
4. The pressed boards are sent to the drilling process, where holes will be drilled in the pressed boards and, on the external surfaces, the circuit pattern will be created.
5. The other steps are the labeling and final inspection processes.

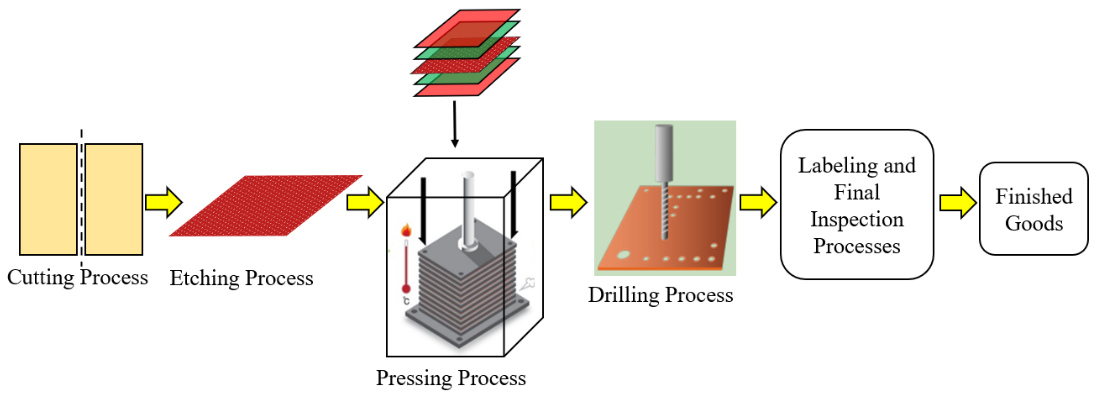


Figure 1.3: The steps of the multi-layer PCB fabrication.

Figure 1.3 summarizes the steps of the multi-layer PCB fabrication. The multi-layer PCB fabrication is complex and composes of several processes. However, this dissertation only focuses on the pressing process. Actually, the pressing process is related to a mathematical optimization since it can be considered as a scheduling problem. The problem description of the pressing process will be explained in Chapter 3.

1.2.2 Linear programming

A linear programming (LP) problem is a class of mathematical optimization problems. An LP problem can be a maximization problem or a minimization problem. The objective of an LP problem is a linear function, and constraints in an LP problem are also linear, which can be linear equalities or inequalities. The variables in an LP problem are continuous. There are two general forms of an LP problem, i.e., standard and canonical forms [4].

An LP problem is in the standard form when all constraints are equalities and all variables are nonnegative. A maximization problem and a minimization problem of LP can be mathematically formulated in the standard form as shown in Table 1.1. Note that a minimization problem in the standard form can be transformed into an equivalent maximization problem in the standard form (and conversely) by multiplying the given objective function by -1, i.e.,

$$\min z = -\max\ -z.$$

Table 1.1: The standard form of maximization and minimization problems of LP.

Maximization problem	**Minimization problem**
$\max \quad z = \sum_{j=1}^{n} c_j x_j$	$\min \quad z = \sum_{j=1}^{n} c_j x_j$
s.t. $\quad \sum_{j=1}^{n} a_{ij} x_j = b_i \quad (i = 1, 2, ..., m)$	s.t. $\quad \sum_{j=1}^{n} a_{ij} x_j = b_i \quad (i = 1, 2, ..., m)$
$x_j \geq 0 \quad (j = 1, 2, ..., n)$	$x_j \geq 0 \quad (j = 1, 2, ..., n)$

A maximization problem or a minimization problem of LP is in canonical form when all constraints are of the $\leq$ type or of the $\geq$ type, respectively, and all variables are nonnegative. A maximization problem and a minimization problem of LP can be

Table 1.2: The canonical form of maximization and minimization problems of LP.

Maximization problem	**Minimization problem**
max $z = \sum_{j=1}^{n} c_j x_j$	min $z = \sum_{j=1}^{n} c_j x_j$
s.t. $\sum_{j=1}^{n} a_{ij} x_j \leq b_i \quad (i = 1, 2, ..., m)$	s.t. $\sum_{j=1}^{n} a_{ij} x_j \geq b_i \quad (i = 1, 2, ..., m)$
$x_j \geq 0 \quad (j = 1, 2, ..., n)$	$x_j \geq 0 \quad (j = 1, 2, ..., n)$

mathematically formulated in the canonical form as shown in Table 1.2.

In the canonical form, a minimization problem can be transformed into a maximization problem (and conversely) by simple manipulations as follows.

- The objective of minimization can be converted to maximization by multiplying the given objective function by -1.
- A given constraint, which is in $\geq$ type, in a minimization problem can be converted to the constraint in $\leq$ type by multiplying the inequality by -1.

In both the standard and canonical forms, all variables must be nonnegative. If a variable x_j in the problem is unrestricted in sign, it can then be replaced by $x'_j - x''_j$, where $x'_j \geq 0$ and $x''_j \geq 0$. Moreover, an LP problem in the canonical form can be converted to the standard form as follows.

- In a maximization problem, a given constraint in $\leq$ type can be converted to an equation form by adding a nonnegative slack variable x_{n+1},

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i \;\rightarrow\; \sum_{j=1}^{n} a_{ij} x_j + x_{n+1} = b_i.$$

- In a minimization problem, a given constraint in $\geq$ type can be converted to an equation form by subtracting a nonnegative surplus variable x_{n+1},

$$\sum_{j=1}^{n} a_{ij}x_j \geq b_i \;\rightarrow\; \sum_{j=1}^{n} a_{ij}x_j - x_{n+1} = b_i.$$

On the other hand, an LP problem in the standard form can be converted to the canonical form as follows.

- If the problem is a maximization problem, an equation of the form $\sum_{j=1}^{n} a_{ij}x_j = b_i$ can be transformed into the following two inequalities,

$$\sum_{j=1}^{n} a_{ij}x_j = b_i \;\rightarrow\; \sum_{j=1}^{n} a_{ij}x_j \leq b_i \text{ and } -\sum_{j=1}^{n} a_{ij}x_j \leq -b_i.$$

- If the problem is a minimization problem, an equation of the form $\sum_{j=1}^{n} a_{ij}x_j = b_i$ can be transformed into the following two inequalities,

$$\sum_{j=1}^{n} a_{ij}x_j = b_i \;\rightarrow\; \sum_{j=1}^{n} a_{ij}x_j \geq b_i \text{ and } -\sum_{j=1}^{n} a_{ij}x_j \geq -b_i.$$

In an LP problem, a set of values of variables $x_1, x_2, ..., x_n$ satisfying all the linear constraints is called a *feasible solution*. The set of all feasible solutions constitutes a *feasible region*. The LP problem aims to find a feasible solution that maximizes or minimizes the objective function, which is called an *optimal solution*. A well-known approach used for solving an LP problem is the simplex method. In addition, if all variables in an LP problem are restricted to be integer, the LP problem becomes an *integer linear programming* (ILP) problem. If all variables are restricted to be only 0 or 1, the problem becomes a *binary integer linear programming* (BILP) problem. The classical solution approaches for solving an ILP and BILP are branch and bound and cutting plane [5].

Furthermore, in an LP problem, if some of the variables are restricted to be integer, the problem is called a *mixed-integer linear programming* (MILP) problem. An integer variable in a MILP problem can be discrete or binary. Let $\mathbb{Z}_0^+$ be the set of all nonnegative integers. A MILP problem can be mathematically formulated in the standard form [5] as follows.

$$\begin{aligned}
\max \quad & z = \sum_{j=1}^{n} c_j x_j + \sum_{k=1}^{p} d_k y_k & \\
\text{s.t.} \quad & \sum_{j=1}^{n} a_{ij} x_j + \sum_{k=1}^{p} g_{ik} y_k = b_i & (i = 1, 2, ..., m) \\
& x_j \geq 0 & (j = 1, 2, ..., n) \\
& y_k \in \mathbb{Z}_0^+ & (k = 1, 2, ..., p)
\end{aligned}$$

The classical solution approaches for solving a MILP problem are also branch and bound as well as cutting plane [5]. Nowadays, MILP is widely used in many application areas, such as scheduling, production planning, and supply chain management.

1.2.3 Nonsimultaneous constraints

In an LP problem, a set of constraints is nonsimultaneous if only some constraints in the set must be satisfied in the problem. Nonsimultaneous constraints do not follow the assumption of LP and MILP because all constraints must be satisfied simultaneously. However, we can convert a problem that includes nonsimultaneous constraints to an equivalent problem which has all simultaneous constraints using binary variables. If the original formulation of the problem fits an LP format except the part of nonsimultaneous constraints, introducing binary variables can transform the problem into a MILP problem.

An example of nonsimultaneous constraints, which can be applied in various situations, is either-or constraints. This constraint type will also be used in our proposed models for the pressing process scheduling in Chapter 3. In either-or constraints, at least one of two constraints must be hold but not necessarily both. A general form of either-or constraints can be expressed as follows.

$$\left.\begin{array}{ll}\text{Either} & f(x_1, x_2, ..., x_n) \leq d_1 \\ \text{or} & g(x_1, x_2, ..., x_n) \leq d_2\end{array}\right\} \tag{1.1}$$

Note that, for an LP problem, the functions f and g in (1.1) are linear. To convert either-or constraints (1.1) to simultaneous constraints, let F_1 and F_2 be the sets of all feasible solutions of the first and the second constraint in (1.1), respectively. Assume that all constraints in (1.1) are bounded in the intersection of F_1 and F_2, i.e., there exists $M_1 \in \mathbb{R}$ such that $\forall(x_1, x_2, ..., x_n) \in F_1 \cap F_2,\ f(x_1, x_2, ..., x_n) \leq M_1$ and there exists $M_2 \in \mathbb{R}$ such that $\forall(x_1, x_2, ..., x_n) \in F_1 \cap F_2,\ g(x_1, x_2, ..., x_n) \leq M_2$. The either-or constraints can be converted to two simultaneous constraints using an auxiliary binary variable y and a large positive number M, where $M \geq \max\{M_1, M_2\}$, as follows.

$$\left.\begin{aligned} f(x_1, x_2, ..., x_n) &\leq d_1 + My \\ g(x_1, x_2, ..., x_n) &\leq d_2 + M(1-y) \end{aligned}\right\} \tag{1.2}$$

From (1.2), if $y = 0$, the second constraint, $g(x_1, x_2, ..., x_n) \leq d_2 + M$, is redundant while the first constraint, $f(x_1, x_2, ..., x_n) \leq d_1$, is satisfied. If $y = 1$, the constraint $f(x_1, x_2, ..., x_n) \leq d_1 + M$ is redundant while the other constraint, $g(x_1, x_2, ..., x_n) \leq d_2$, is satisfied. Hence, the either-or constraints (1.1) can be added to an LP problem using the form (1.2). The new problem becomes MILP due to the auxiliary binary variable y.

A following example illustrates how to reformulate an LP problem with either-or constraints into a MILP problem.

Example 1.2.1. Consider the LP problem

$$\begin{aligned} \max \quad & 50x_1 + 60x_2 & \\ \text{s.t.} \quad & 2x_1 + x_2 \leq 300 & (1.3) \\ \text{Either} \quad & 3x_1 + 4x_2 \leq 509 & (1.4) \\ \text{or} \quad & 4x_1 + 7x_2 \leq 812 & (1.5) \\ & x_1 \geq 0, x_2 \geq 0. & \end{aligned}$$

Let M be a large positive number and y be a binary variable. This LP problem can be reformulated into a MILP problem (P_1) as follows.

$$
\begin{array}{lll}
(P_1) & \max & 50x_1 + 60x_2 \\
 & \text{s.t.} & 2x_1 + x_2 \le 300 \\
 & & 3x_1 + 4x_2 \le 509 + My \\
 & & 4x_1 + 7x_2 \le 812 + M(1-y) \\
 & & x_1 \ge 0, x_2 \ge 0, y \in \{0,1\}.
\end{array}
$$

A possible value of M for Problem P_1 can be chosen as follows. From Constraints (1.3) – (1.5), if $x_2 = 0$, the largest value of x_1 is $\max\{\frac{300}{2}, \frac{509}{3}, \frac{812}{4}\} = 203$. On the other hand, if $x_1 = 0$, the largest value of x_2 is $\max\{300, \frac{509}{4}, \frac{812}{7}\} = 300$. From either-or constraints (1.4) – (1.5), the maximum value of the left hand side (LHS) of Constraint (1.4) is $3x_1 + 4x_2 \le 3(203) + 4(300) = 1809 =: M_1$, while the maximum value of LHS of Constraint (1.5) is $4x_1 + 7x_2 \le 4(203) + 7(300) = 2912 =: M_2$. Let $M = \max\{M_1, M_2\} = \max\{1809, 2912\} = 2912$. This value can be a possible value of M for Problem P_1.

1.2.4 Heuristic

In a mathematical optimization, a heuristic is a technique designed for solving a problem more quickly when classical methods cannot find an optimal solution within an acceptable time. Many optimization problems from the real world are very large and are not easy to find an optimal solution. Therefore, heuristic algorithms are preferred tools for finding an approximated solution of an optimization problem within a reasonable time. The solution from a heuristic algorithm may not be the best of all solutions to the problem, but it is acceptably good. In general, a heuristic algorithm will be designed upon a problem at hand.

In this section, two examples of heuristics for solving the parallel machine scheduling problem (PMSP) [6] are explained. The problem description of PMSP is as follows:

- There are m identical parallel machines.
- There are n jobs to be processed, where each job requires a single operation and can be processed on anyone of the m machines or on anyone that belongs to a given

subset of the m machines.

- The objective of PMSP is to minimize the maximum completion time over all jobs, which is called makespan ($C_{\max}$), i.e., $C_{\max} = \max\{C_1, C_2, ..., C_n\}$ where C_j is the completion time of job $j \in \{1, 2, ..., n\}$.

Two heuristics for solving PMSP that are explained here include the longest processing time (LPT) and load balancing. The idea of LPT and load balancing are as follows.

1. LPT: All jobs are arranged in descending order of processing times. The jobs that have large values of processing times are given high priority for scheduling on the parallel machines. The m longest jobs are firstly assigned to the m machines. The longest job among the unprocessed jobs is then assigned to the machine that can start the job as fast as possible until all jobs are scheduled.

2. Load balancing: This heuristic tries to balance load in each machine. Firstly, the tentative load per machine can be computed from $\frac{T_w}{m}$, where T_w is the total processing time of all jobs. Then, all jobs are partitioned into m groups, where the total processing time of all jobs in each group must be equal to or similar to the tentative load per machine.

A following example [6] illustrates how to solve PMSP using the LPT and load balancing heuristics.

Example 1.2.2. Consider PMSP with 9 jobs ($J_1, J_2, ..., J_9$) and 4 machines (M_1, M_2, M_3, M_4), where the processing time of each job (p_j) is shown in the following table. Solve this PMSP using the LPT and load balancing heuristics.

Job	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8	J_9
p_j	7	7	6	6	5	5	4	4	4

Solution using LPT

- Firstly, all jobs are sequenced in descending order of processing times. Because jobs in the table have already been arranged according to processing times, we get the LPT sequence as $J_1 - J_2 - J_3 - J_4 - J_5 - J_6 - J_7 - J_8 - J_9$.

- Next, jobs J_1 – J_4 are assigned to machines M_1 – M_4, respectively, as shown in Table 1.3. These jobs can be started at time 0 on these machines because each job is firstly scheduled on each machine.

- Job J_5 is then selected to be scheduled. It is assigned to a machine that can start this job as early as possible, which can be machines M_3 or M_4. Assume that job J_5 is assigned to process on machine M_3 at time 6. Similarly, the next job (J_6) will be started at time 6 in machine M_4, as shown in Table 1.3.

- The next two unscheduled jobs, which are J_7 and J_8, will be scheduled on machine M_1 and M_2 at time 7, respectively, as shown in Table 1.3.

- The last job or J_9 can be scheduled on any machine because it can be started as early as possible at time 11 on every machine. Assume that job J_9 is scheduled on machine M_1 as shown in Table 1.3. We get a solution of this PMSP because all jobs are already scheduled. Figure 1.4 shows the Gantt chart of this solution, and the makespan of this solution is 15.

Table 1.3: A feasible schedule for PMSP using LPT heuristic.

Job	**Machine**	**Starting time (S_j)**	**Processing time (p_j)**	**Completion time (C_j)**
J_1	M_1	0	7	7
J_2	M_2	0	7	7
J_3	M_3	0	6	6
J_4	M_4	0	6	6
J_5	M_3	6	5	11
J_6	M_4	6	5	11
J_7	M_1	7	4	11
J_8	M_2	7	4	11
J_9	M_1	11	4	15

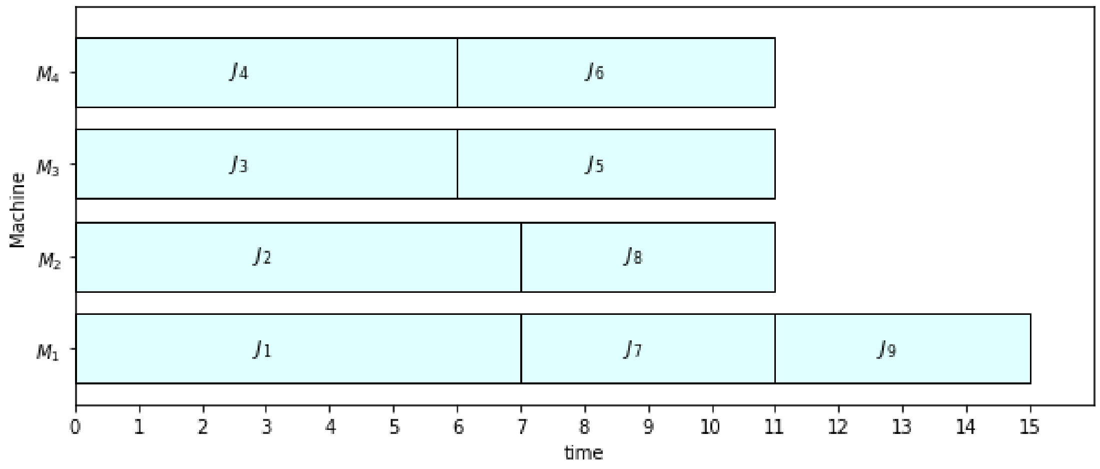


Figure 1.4: Gantt chart of the solution of PMSP using LPT heuristic.

Solution using load balacing

In this problem, the tentative load per machine is $\frac{T_w}{m} = \frac{48}{4} = 12$. Then, all 9 jobs are partitioned into 4 groups, where the total processing time of all jobs in each group must be equal to or similar to 12. One possible combination is shown in Table 1.4. Figure 1.5 shows the Gantt chart of this solution. The makespan of this solution is 12, which is better than the makespan of the solution from LPT heuristic.

Table 1.4: A feasible schedule for PMSP using load balancing heuristic.

Machine	**Job group**	**Total processing time in the group**
M_1	$\{J_1, J_5\}$	$7+5=12$
M_2	$\{J_2, J_6\}$	$7+5=12$
M_3	$\{J_3, J_4\}$	$6+6=12$
M_4	$\{J_7, J_8, J_9\}$	$4+4+4=12$

1.2.5 Flexible job-shop scheduling problem

A flexible job-shop scheduling problem (FJSP) is a type of scheduling problems which uses assignment and sequencing to obtain an optimal solution. This problem is related to the pressing process scheduling problem that will be considered in this dissertation because they utilize the same techniques (assignment and sequencing) for solving the problem. The problem description of FJSP is as follows [7].

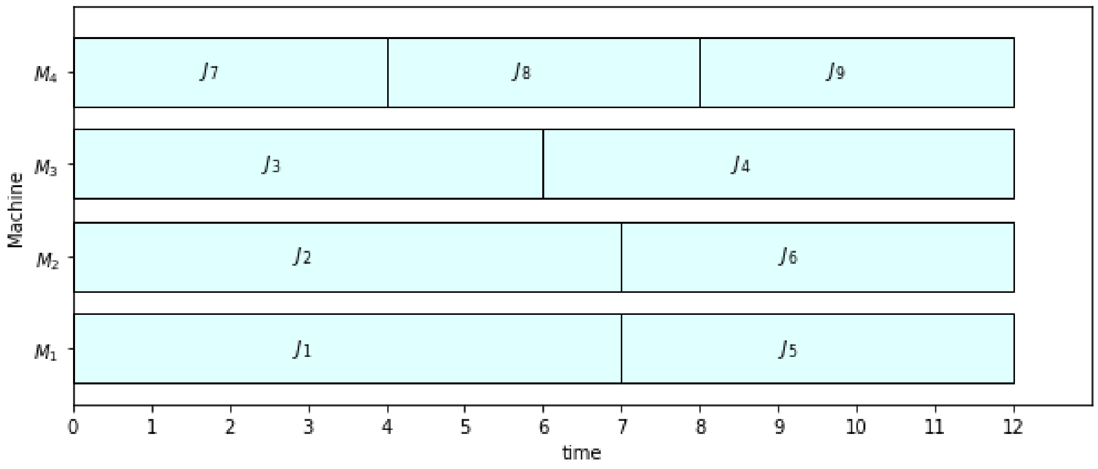


Figure 1.5: Gantt chart of the solution of PMSP using load balancing heuristic.

- This problem has n jobs $(J_1, J_2, ..., J_n)$ and m machines $(M_1, M_2, ..., M_m)$.
- Each job J_i, where $i \in \{1, 2, ..., n\}$, consists of a sequence of operations $(O_{i1}, O_{i2}, ...O_{in_i})$ to be processed to complete the job.
- Each operation O_{ij}, where $j \in \{1, 2, ..., n_i\}$, can be processed on any machine or on anyone that belongs to a given subset of the m machines.
- The processing time of each operation is known and depends on a machine.
- Each machine can process only one operation at a time.
- The problem is to assign each operation to a suitable machine (assignment) and to sequence the operations on the machines (sequencing) so that the makespan $(C_{\max})$ is minimized.

An example of FJSP with 3 jobs and 4 machines [8] is shown in Table 1.5. There are two, three, and two operations in J_1, J_2, and J_3, respectively. The number in each cell represents the processing time of that operation on the corresponding machine. The "-" in this table means that the machine cannot process the corresponding operation.

A feasible solution or feasible schedule of this problem is shown in Figure 1.6. In this solution, operations O_{11}, O_{12}, and O_{21} are assigned to machine M_1, whereas operations

Table 1.5: An example of FJSP with 3 jobs and 4 machines.

Job	Operation	Machine			
		M_1	M_2	M_3	M_4
J_1	O_{11}	4	7	6	5
	O_{12}	2	6	-	5
J_2	O_{21}	4	5	7	-
	O_{22}	5	-	6	3
	O_{23}	-	5	4	7
J_3	O_{31}	5	3	-	6
	O_{32}	-	-	4	-

O_{32} and O_{23} are assigned to machine M_3. Besides, operations O_{31} and O_{22} are assigned to machines M_2 and M_4, respectively. The sequencing of operations on each machine is also illustrated in Figure 1.6. In this figure, we can see that all operations in each job are processed according to the sequence of ordered operations. Furthermore, each machine executes only one operation at a time, i.e., any two operations do not overlap. The makespan of this solution is 17. Note that, in Figure 1.6, the same color represents the operations of the same job.

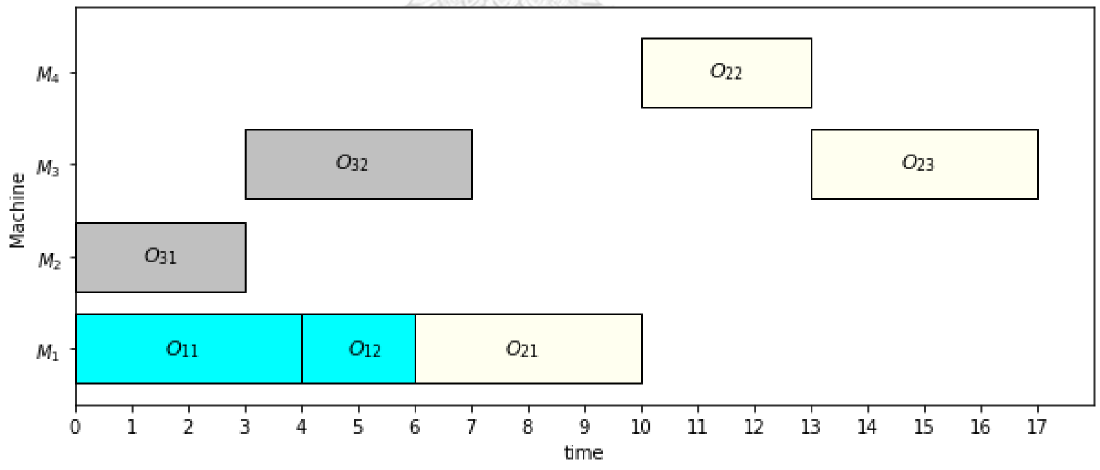


Figure 1.6: Gantt chart of a solution of the FJSP example.

FJSP has been comprehensively studied in literature. Many techniques have been presented to solve FJSP, such as a mathematical model [9, 10], a genetic algorithm (GA) [7, 11], a hybrid of artificial immune and simulated annealing (AISA) algorithm [10], and a hybrid of GA and tabu search algorithm [12].

1.2.6 Performance measures for comparing MILP models

Normally, a mathematical model is the initial step to study an optimization problem [13], especially a new one. Mathematical programming models have commonly been proposed for finding an optimal solution for a scheduling problem, in particular MILP models [14]. This is because a MILP model can clearly explain the constraints, objective, and the specifications of the scheduling problem. It can also be used as an idea to design a heuristic algorithm for solving the problem [15]. Furthermore, MILP is a type of optimization model that allows an exact method, such as branch and bound and cutting plane, to solve an optimization problem and provides an optimal solution if one exists. In genaral, if an optimization problem can be formulated in MILP, there always exist many alternative formulations of MILP for representing the optimization problem [5]. One alternative may be easier to solve than others. Therefore, there must be some criteria for comparing their efficiency.

There are many factors that affect the efficiency of a MILP model. According to Meng et al. [16], the number of binary variables (NBV), the number of constraints (NC), the number of continuous variables (NCV), the dimensionality of decision variables, and the constraints' tightness all affect the performance of a MILP model. Furthermore, the coefficient of decision variables is another factor because it affects the quality of the LP relaxation of the MILP model [5]. NBV is the most important factor that influences the model's performance, and NC is the next most important factor [17].

There are two common performance measures of size complexity and computational complexity [13,16,18,19] that are used to evaluate the performance of any two alternative MILP models. The size complexity is measured by counting NBV, NCV, and NC, which are generated by the MILP model. If NBV, NCV, and NC in a model are less than NBV, NCV, and NC in another model, respectively, the first one is absolutely better than the second one in terms of the size complexity. However, the performance of a MILP model does not only depend on the model size because there are other factors such as the constraints' tightness and the coefficient of decision variables. Many researchers also evaluate the performance of MILP models in terms of the computational complexity, which is measured by the computational time for solving the problem. If a MILP model can solve the problem faster than another MILP model, the first one is better than the second one in terms of the computational complexity. When a MILP model, says the first model, outperforms another model, says the second model, in terms of both size and computational complexities, it can be concluded that the first model is better than the second model.

1.3 Research Objectives

This dissertation focuses on the pressing process scheduling in multi-layer PCB manufacturing. The objectives of this dissertation are summarized as follows.

1. This work aims to propose a MILP model (hereafter Model 1) for the pressing process scheduling in multi-layer PCB manufacturing with the objective of minimizing the makespan.
2. This work aims to propose an alternative MILP model of the same problem (Model 2), which is an improvement of Model 1.
3. This work aims to present a heuristic algorithm (three-phase-PCB-pressing heuristic algorithm or 3P-PCB-PH algorithm) for solving the pressing process scheduling.

In this work, some actual data of the pressing process were acquired from a PCB company. The test problems in this work are generated from the actual data and are solved using all the proposed methods to compare their performances.

1.4 Research Contributions

The contributions of this dissertation can be summarized as follows.

1. To the best of the authors' knowledge, this work is the first research study on the

scheduling of the pressing process, which is a real application from PCB manufacturing industry and has never been investigated in literature.

2. This work proposes two novel MILP models for solving the pressing process scheduling, i.e., Models 1 and 2. In this work, Model 2 is an improved version of Model 1, where Model 2 is superior than Model 1 in terms of both size and computational complexities.

3. Because of the complication of the pressing process scheduling, a 3P-PCB-PH algorithm for solving this problem is also proposed. The 3P-PCB-PH algorithm can find a good solution to the pressing process scheduling within a reasonable amount of time.

4. The proposed Models 1 and 2 as well as the 3P-PCB-PH algorithm can be options to find an optimal schedule or a high quality schedule for any PCB manufacturer to reduce their production time and cost.

1.5 Overview of Dissertation

This dissertation contains five chapters and is organized as follows. Chapter 1 provides the introduction of this research study, which includes motivation, background knowledge, research objectives, and research contributions. The background knowledge consists of multi-layer PCB fabrication, LP, nonsimultaneous constraints, heuristic, FJSP, and performance measures for comparing alternative MILP models. Chapter 2 is the literature review which is related to processes in multi-layer PCB manufacturing as well as improvement and comparison of the performance of MILP models of various scheduling MILP models. Chapter 3 explains the problem description and the proposed methodology, which are divided into four sections. In the first section, the pressing process scheduling is described in detail. The proposed MILP models, which are Models 1 and 2, are presented in the second and third sections, respectively. In this work, Model 2 is improved from Model 1, where it has a smaller model size and uses less computational time than Model 1 for solving the pressing process scheduling problem. The last section proposes the 3P-

PCB-PH algorithm. Numerical examples are shown in Chapter 4. Finally, the conclusions of this research study are in Chapter 5.

CHAPTER II

LITERATURE REVIEW

The literature review of this dissertation is divided into two parts. The first part reviews the related studies about processes in multi-layer PCB manufacturing, and the second part reviews the related studies about improving and comparing the performance of MILP models for various types of scheduling problems.

2.1 Literature review about processes in multi-layer PCB manufacturing

As stated in Chapter 1, multi-layer PCB manufacturing has four stages, i.e., the design, the fabrication, the assembly, and the testing. There are many research studies that are related to the fabrication, the assembly, and the testing. In the multi-layer PCB fabrication, there are many time- and cost-consuming processes, such as cutting and drilling processes. Every PCB manufacturer wants to keep the waste areas from laminate cutting to a minimum in the cutting process. The two-dimensional cutting stock problem (2DCSP), which is a well-known problem in optimization, can represent this process. In the drilling process, every PCB company wishes to determine an optimal path for drilling the hole in the specified positions in the circuit pattern in order to reduce the travel time of the drilling device. An optimization problem that is related to the drilling process is the drilling path optimization problem (DPOP).

2DCSP has been extensively studied in the literature. The first mathematical model for 2DCSP was presented by Gilmore and Gomory [20], and the model was solved via a column generation technique. An exact arc-flow model, which is an ILP formulation, for 2DCSP with two stages and guillotine cutting constraints was presented in [21]. An exact branch-and-price algorithm was also proposed for solving 2DCSP with two stages and guillotine cutting constraints [22]. For 2DCSP with multiple stock sizes, several heuristic algorithms based on column generation were developed such as [23,24]. In addition, some research studies on the cutting process used real data from PCB manufacturers [25, 26]. In [25], many appropriate heuristic algorithms were presented for solving 2DCSP with multiple stock sizes using real problems from a PCB company. Moreover, a mathematical model for 2DCSP with fixed size usable leftover was proposed and solved using a column generation technique with real instances from a PCB company [26].

DPOP can be modeled as a traveling salesman problem (TSP), which is a very well-known problem in optimization. In drilling process, the drilling device should be steered to the position of each hole exactly once in order to minimize the total travel time of the drilling device. Numerous heuristic techniques have been proposed for solving DPOP, such as GA [27], an ant colony system [28], a particle swarm optimization (PSO) [29], a cuckoo search algorithm [30, 31], and a hybridized cuckoo search-genetic algorithm [32].

The PCB assembly is a time-consuming process in the manufacturing of PCB that involves setting electronic components at the predefined places on a bare board. Normally, there are a lot of placement machines in a PCB assembly line. Because of many configurations and types of the placement machines, several machines can have different unit assembly times for the same component. Each bare PCB goes through all machines to finish the placement of components. Thus, the components must be served to suitable machines in order to minimize the assembly time. This conducts the workload balance problem in the PCB assembly line. The goal of this problem is to minimize the maximum assembly time among all placement machines in the line for a specific PCB type, which is called the cycle time of the assembly line. GA [33] and a branch-and-bound-based optimization algorithm [34] have been presented for solving this problem. Moreover, a MILP formulation for a workload balance problem in the PCB assembly line with additional constraints, such as precedence constraints between components and the use of feeder modules, was proposed in [35]. Furthermore, a hierarchical heuristic for solving the integrated workload balancing and the single machine optimization problem was presented in [36].

The testing process is the final stage in the PCB manufacturing. Before PCBs are used in the field, environmental stress screening chambers are regularly used on PCBs to inspect primal fallouts. A chamber can test many PCBs at the same time, i.e., PCBs can be processed in batches. The processing time of a batch is the longest processing time among all jobs that constitute the batch. Consequently, the PCB testing process can be considered as a batch processing machine scheduling problem (BPMSP), which has been widely studied in literature. In [37], a simulated annealing (SA) algorithm was presented for solving a single BPMSP with the objective of minimizing the makespan. Reference [38] proposed a PSO algorithm for solving a nonidentical parallel BPMSP with the objective of minimizing the makespan. A MILP model, which has the objective of minimizing the makespan, was proposed for the flow-shop scheduling problem (FSP) with batch processing machines [39]. Moreover, a simulation-based intelligence optimization method was proposed for solving FSP with multiple heterogeneous batch processing machines and the objective of minimizing the makespan [1]. Furthermore, a nonidentical parallel BPMSP with the objective of minimizing the total weighted tardiness was considered in [40], and the problem was solved using a PSO algorithm.

2.2 Literature review about improving and comparing the performance of MILP models for various types of scheduling problems

In general, a scheduling problem is formulated in a form of a MILP model [14]. Since there always exist many, possibly infinite, alternative formulations for a given integer programming problem [5], many researchers aim to improve MILP models so that they can solve scheduling problems more effectively.

There are many research studies that aim to compare and improve the performance of MILP models for various types of a scheduling problem. In [41], the authors considered the flow-shop, permutation flow-shop, job-shop, and open-shop scheduling problems (FSP, PFSP, JSP, OSP) with limited waiting time constraints and the objective of minimizing the makespan. For each problem, the authors presented two MILP models and determined the best model by comparing the size complexity between them. Reference [42] considered OSP with the objective of minimizing the total tardiness. The authors proposed four

MILP models for OSP and determined the best model by comparing the performance in terms of both size and computational complexities. In [43], a MILP model for FJSP with sequence dependent setup time (SDST-FJSP) and the objective of minimizing the toal tardiness was presented. The proposed MILP model was shown that it outperformed the existing model for SDST-FJSP from [44] in terms of both size and computational complexities. Besides, two MILP models for FJSP with the objective of minimizing the makespan were proposed in [10]. Both MILP models were evaluated the performance with the existing models for FJSP [9,45,46]. The results showed that they are better than the existing models [9,45,46] in terms of both size and computational complexities.

As for the hybrid flow-shop scheduling problem (HFSP), four MILP models with the objective of minimizing the makespan were proposed in [18]. The authors determined the best model for HFSP by comparing the performance of these models in terms of both size and computational complexities. In [13], the distributed job-shop scheduling problem (DJSP) was considered, where the objective was to minimize the makespan. A MILP model for DJSP was proposed and shown that it outperformed the existing model for DJSP from [47] in terms of both size and computational complexities. In addition, reference [16] studied HFSP with unrelated parallel machines (UPM-HFSP) and the objective of minimizing the makespan. Eight MILP models for this problem were proposed, and the authors evaluated the performance of these models with the existing model from [48] to determine the best model. Moreover, distributed FJSP (DFJSP) with the objective of minimizing the makespan was studied in [19]. Four MILP models for DFJSP were proposed and evaluated the performance in terms of both size and computational complexities to determine the best model for DFJSP.

After extensively reviewing the literature, we can see that many processes in multilayer PCB manufacturing are investigated, but there is no research study on the pressing process. Therefore, the pressing process is a new application, and this dissertation focuses on this process. The goals of this dissertation are to provide effective approaches for solving the pressing process scheduling. Two MILP models with the objective of minimizing the makespan for the pressing process scheduling are presented, and these models will be compared the performance in terms of both size and computational complexities. Note that the objective of minimizing the makespan can imply a good utilization of available resources [49], which is generally the main objective of any PCB manufacturer. Furthermore, a heuristic algorithm for solving the pressing process scheduling is also presented. The details of all the proposed approaches are in Chapter 3.

CHAPTER III

PROBLEM DESCRIPTION AND METHODOLOGY

This chapter is divided into four sections. Section 1 describes the problem description of the pressing process scheduling. Sections 2 – 4 present the proposed approaches for solving the pressing process scheduling, which include Model 1, Model 2, and the 3P-PCB-PH algorithm, respectively.

3.1 Problem description of the pressing process scheduling

This section explains the pressing process in multi-layer PCB manufacturing. The aim of the pressing process is to press the stack of core, prepreg, and copper foil to form a multi-layer board. In this study, the stack is called a panel as shown in Figure 3.1. The details of the pressing process are as follows.

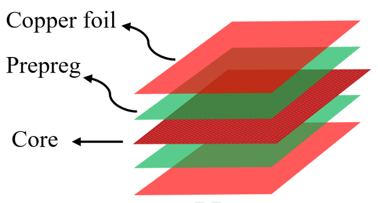


Figure 3.1: An example of a panel.

- A certain number of panel types (I) and the demand of each panel type ($d_i; i \in \{1, 2, ...I\}$) are given.
- The company has (K) sizes of stainless-steel templates (SSTs) and L layouts. A layout is a pattern of panel arrangement on SST. The result from a placement of panels on SST is called a *book*.
- To arrange panels on SST, outer and inner gaps are required. The outer gap (G) is the minimal gap between each panel and the SST's edges, and the inner gap (g)

is the minimal gap between two panels in a book as shown in Figure 3.2. Both inner and outer gaps depend on each panel type. The number of panels on a book depends on these gaps, the SST size, the panel size, and the layout. Generally, a PCB manufacturer has its own formula for calculating the number of panels on a book using SST and a layout.

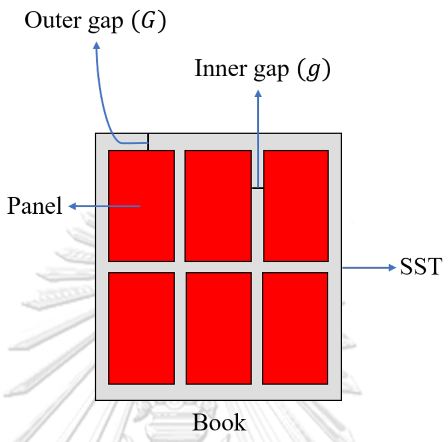


Figure 3.2: Inner and outer gaps.

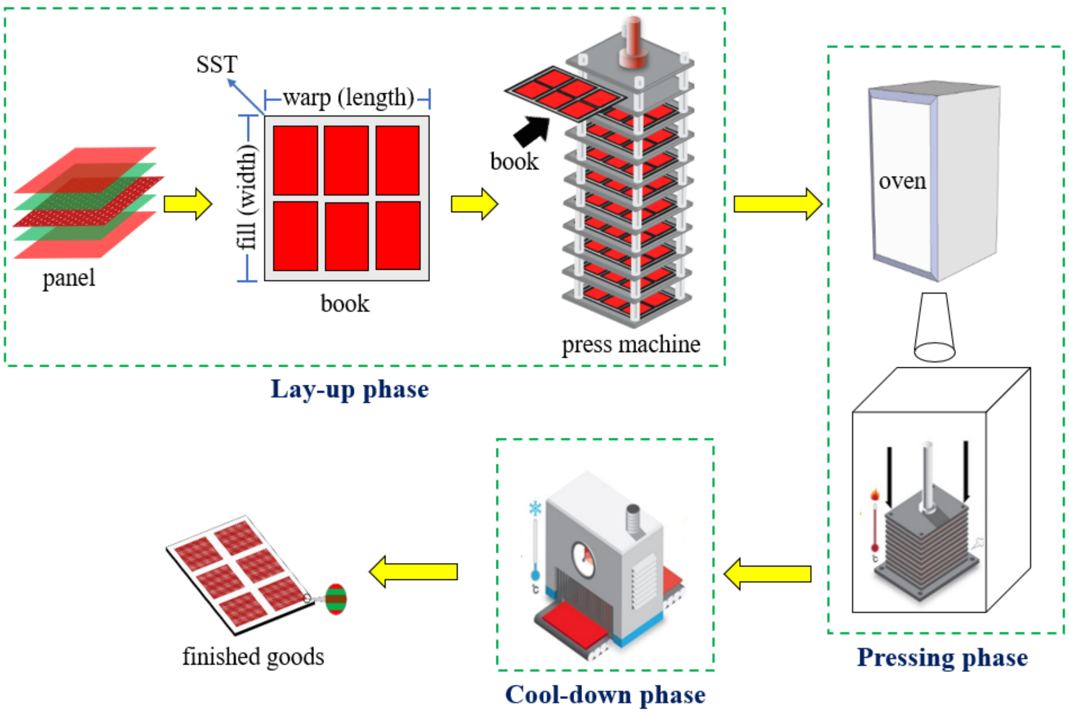


Figure 3.3: Processes in one cycle of a press machine.

- A cycle of a press machine in the pressing process has three phases as shown in Figure 3.3, which consist of the following:

    1. Lay-up phase: A number of panels of the same type, a SST size, and a layout are used to create books. Then, the books are loaded into all openings (slots) of the press machine.
    2. Pressing phase: The press machine which is completely inserted with books is put into an oven, where the books are pressed and heated. The press machine is taken out of the oven after finishing the press.
    3. Cool-down phase: In the press machine, the pressed books are cooled down. Lastly, the books are removed from the press machine to finish a press machine cycle.

- Each phase in the pressing process has a certain processing time.
- All three phases of a cycle of a press machine must be processed without interruption.
- After a press machine has completed a cycle, it can instantaneously start the next cycle. For an oven, after the current pressing phase is done, it is immediately available for the pressing phase of another press machine.
- A certain number of ovens (O) and press machines (P) are given, where each press machine has m openings.
- A planning horizon of the pressing process scheduling is considered, such as 3 days, where the maximum number of available cycles (T) of each press machine to operate the pressing process is given. Actually, the production planning department of the PCB company can approximate this value from the resources and the order of the customer.

Moreover, this work has the following assumptions:

1. For each SST size, the number of SSTs is unlimited.

2. Within the planing horizon and available resources, the demands of panels from the customers can be met, i.e., the demands or inputs from the customers yield a feasible schedule.

3. This work considers the problem in a simple case, where the processing time of each phase in the pressing process is the same (n).

The constraints in pressing process scheduling are as follows:

1. In a cycle of a press machine, all books that are inserted in all openings must have the same patterns, i.e., all books are created using the same panel type, the same SST size, and the same layout. This constraint is required to make sure that the pressure from the press machine can be distributed equally over each panel on the book.

2. Only one of the pressing phase of a press machine can be performed in an oven at a time.

3. The number of finished goods or outputs of each panel type must satisfy the demand.

The objective of the pressing process is to minimize the makespan, which is the maximum completion time of all press machine cycles that operate the pressing process.

3.2 Proposed MILP model (Model 1)

This section proposes the first MILP model, which is called Model 1, for scheduling the pressing process as explained in the previous section. The following notations are used in the formulation of Model 1.

Indices:

i	The index of panel types.
k	The index of sizes of SST.
l	The index of layouts.
p	The index of press machines.
o	The index of ovens.
t	The index of cycles of a press machine.

Parameters:

I	The number of panel types.
K	The number of sizes of SST.
L	The number of layouts.
P	The number of press machines.
O	The number of ovens.
T	The maximum number of available cycles of each press machine.
m	The number of openings of each press machine.
n	The processing time of each phase in the pressing process, i.e., the lay-up, pressing, and cool-down phases.
a_{ikl}	The number of panels of type i per book (or per opening) using stainless size k and layout l. If $a_{ikl} = 0$, it means that panel type i is not compatible with SST size k and layout l.
d_i	The demand of panel type i.
M	A large positive number.

Sets:

$\hat{I}$	The set of all panel types, $\hat{I} = \{1, 2, ..., I\}$.
$\hat{K}$	The set of all sizes of SST, $\hat{K} = \{1, 2, ..., K\}$.
$\hat{L}$	The set of all layouts, $\hat{L} = \{1, 2, ..., L\}$.
$\hat{P}$	The set of all press machines, $\hat{P} = \{1, 2, ..., P\}$.
$\hat{O}$	The set of all ovens, $\hat{O} = \{1, 2, ..., O\}$.
$\hat{T}$	The set of all number of available cycles of each press machine, $\hat{T} = \{1, 2, ..., T\}$.

Decision variables:

x_{iklpt}	Binary variable which is equal to 1 if panel type i is assigned with SST size k and layout l to press machine p at cycle t.
X_{pto}	Binary variable which is equal to 1 if press machine p at cycle t is sent into oven o for operating the pressing phase.
$Y_{ptp't'o}$	Binary variable which is equal to 1 if press machine p at cycle t precedes press machine p' at cycle t' in oven o.
A_{pt}	Continuous variable representing the starting time of the lay-up phase of press machine p at cycle t.
B_{pto}	Continuous variable representing the starting time of the pressing phase of press machine p at cycle t in oven o.
C_{pt}	Continuous variable representing the completion time of press machine p at cycle t.
D_{pto}	Continuous variable representing the completion time of the pressing phase of press machine p at cycle t in oven o.
C'_{pt}	The auxiliary continuous variable, which is equal to C_{pt} if a panel type, a SST size, and a layout are assigned to press machine p at cycle t. Otherwise, it is equal to 0. The necessity of this variable will be explained later.
$C_{\max}$	Continuous variable representing the makespan, which is the maximum completion time of the last cycle of all press machines that operate the pressing process.

The proposed Model 1 can be stated as follows.

Objective:

$$\min \quad C_{\max} \tag{3.1}$$

The objective function (3.1) is to minimize the makespan of the overall process.

Subject to the following constraints:

1. Panel-SST-layout assignment constraint:

$$\sum_{i=1}^{I}\sum_{k=1}^{K}\sum_{l=1}^{L} x_{iklpt} \leq 1,\ \forall p \in \hat{P}, \forall t \in \hat{T} \tag{3.2}$$

Constraint (3.2) makes sure that at most one panel type, one SST size, and one layout can be assigned in each press machine cycle. When a panel type, a SST size, and a layout are assigned to a cycle of a press machine, it means that all books that are loaded into all openings in this cycle have the same pattern. On the other hand, if this press machine at this cycle has no assignment of these materials, it means that this press machine cycle is empty or does not do any work.

2. Panel-SST-layout compatibility constraint:

$$x_{iklpt} \leq a_{ikl},\ \forall i \in \hat{I}, \forall k \in \hat{K}, \forall l \in \hat{L}, \forall p \in \hat{P}, \forall t \in \hat{T} \tag{3.3}$$

Constraint (3.3) ensures that if panel type i cannot be used with SST size k and layout l ($a_{ikl} = 0$), then this pattern cannot be assigned to any cycle of a press machine. This is because if $a_{ikl} = 0$, for some $i \in \hat{I}, k \in \hat{K}, l \in \hat{L}$, then variable x_{iklpt}, for all $p \in \hat{P}, t \in \hat{T}$ on LHS is equal to 0.

3. Demand constraint:

$$\sum_{k=1}^{K}\sum_{l=1}^{L}\sum_{p=1}^{P}\sum_{t=1}^{T} x_{iklpt}(m \cdot a_{ikl}) \geq d_i,\ \forall i \in \hat{I} \tag{3.4}$$

Constraint (3.4) makes sure that the total outputs of each panel type must satisfy the demand. Note that the value $m \cdot a_{ikl}$ on LHS is the total outputs of panels of type i (using SST size k and layout l) from one cycle of a press machine, and the term $\sum\limits_{k=1}^{K}\sum\limits_{l=1}^{L}\sum\limits_{p=1}^{P}\sum\limits_{t=1}^{T} x_{iklpt}(m \cdot a_{ikl})$ represents the total outputs of panels of type i from all cycles and all press machines after finishing the overall process.

4. Constraint for arranging the working cycles in sequential order:

$$\sum_{i=1}^{I}\sum_{k=1}^{K}\sum_{l=1}^{L} x_{iklp(t-1)} \geq \sum_{i=1}^{I}\sum_{k=1}^{K}\sum_{l=1}^{L} x_{iklpt},\ \forall p \in \hat{P}, \forall t \in \hat{T} - \{1\} \tag{3.5}$$

Constraint (3.5) enforces that a panel type, a SST size, and a layout must always be assigned to sequential cycles (starting from cycle 1) of a press machine if possible. In other words, this constraint helps push all empty cycles to be appeared after non-empty cycles (the cycles that have a panel-SST-layout assignment or the cycles that really perform the pressing process). This is because if cycle $t-1$ of press machine p has no panel-SST-layout assignment $\left(\sum_{i=1}^{I}\sum_{k=1}^{K}\sum_{l=1}^{L} x_{iklp(t-1)} = 0\right)$, then, from Constraint (3.5), the right hand side (RHS) term $\sum_{i=1}^{I}\sum_{k=1}^{K}\sum_{l=1}^{L} x_{iklpt} = 0$ for any cycle that is after cycle $t-1$. This means that any cycle t' of press machine p that has a panel-SST-layout assignment $\left(\text{i.e., } \sum_{i=1}^{I}\sum_{k=1}^{K}\sum_{l=1}^{L} x_{iklpt'} = 1\right)$ must be before cycle $t-1$.

5. Press machine assignment constraint:

$$\sum_{o=1}^{O} X_{pto} = 1, \forall p \in \hat{P},\ \forall t \in \hat{T} \tag{3.6}$$

Constraint (3.6) ensures that each press machine cycle must be only assigned to one oven to do the pressing phase.

6. Constraint for setting times of the pressing phase of a cycle where $X_{pto} = 0$:

$$B_{pto} + D_{pto} \leq (X_{pto})M,\ \forall p \in \hat{P}, \forall t \in \hat{T}, \forall o \in \hat{O} \tag{3.7}$$

Constraint (3.7) assures that the starting and end times of the pressing phase of press machine p at cycle t in oven o can be any non-negative value (due to the large positive number M in RHS) if press machine p at cycle t is assigned to do the pressing phase in oven o ($X_{pto} = 1$). Otherwise ($X_{pto} = 0$), these values are set to be 0 since RHS of Constraint (3.7), which is equal to 0, will force both values of B_{pto} and D_{pto} on LHS to be 0.

7. Precedence constraint of cycles of a press machine:

$$A_{pt} \geq C_{p,t-1},\ \forall p \in \hat{P}, \forall t \in \hat{T} - \{1\} \tag{3.8}$$

Constraint (3.8) ensures that any press machine cycle can be started if and only if the previous cycle has been done.

8. Constraint for setting the starting time of the pressing phase of any press machine cycle:

$$\sum_{o=1}^{O} B_{pto} = A_{pt} + n,\ \forall p \in \hat{P}, \forall t \in \hat{T} \tag{3.9}$$

Constraint (3.9) enforces the starting time of the pressing phase of press machine p at cycle t in its assigned oven, say oven o', to be equal to the starting time of this cycle (A_{pt}) plus the processing time n required in the lay-up phase. It should be noted that the term $\sum_{o=1}^{O} B_{pto}$ on LHS is equal to the starting time of the pressing phase of press machine p at cycle t in oven o' because the value of B_{pto} for all $o \in \hat{O} - \{o'\}$ is equal to 0 from Constraint (3.7).

9. Constraint for setting the completion time of any press machine cycle:

$$C_{pt} = A_{pt} + 3n,\ \forall p \in \hat{P}, \forall t \in \hat{T} \tag{3.10}$$

Constraint (3.10) represents that the completion time of press machine p at cycle t (C_{pt}) is equal to the starting time of this cycle (A_{pt}) plus the processing time $3n$, which is the processing time of one cycle.

10. Constraint for setting the completion time of the pressing phase of any press machine cycle:

$$(B_{pto} + n) - (1 - X_{pto})M \leq D_{pto},\ \forall p \in \hat{P}, \forall t \in \hat{T}, \forall o \in \hat{O} \tag{3.11}$$

$$D_{pto} \leq (B_{pto} + n) + (1 - X_{pto})M,\ \forall p \in \hat{P}, \forall t \in \hat{T}, \forall o \in \hat{O} \tag{3.12}$$

Constraints (3.11) and (3.12) make sure that if press machine p at cycle t is assigned to perform the pressing phase in oven o ($X_{pto} = 1$), the end time of the pressing phase of press machine p at cycle t in oven o (D_{pto}) is equal to the starting time (B_{pto}) plus the processing time n required in this pressing phase ($D_{pto} = B_{pto}+n$). Otherwise ($X_{pto} = 0$), these constraints are redundant.

11. Constraint for avoiding an overlap in an oven:

$$B_{pto} \geq D_{p't'o} - (Y_{ptp't'o})M, \ \forall p, p' \in \hat{P}, \ p \neq p', \ \forall t, t' \in \hat{T}, \ \forall o \in \hat{O} \tag{3.13}$$

$$B_{p't'o} \geq D_{pto} - (1 - Y_{ptp't'o})M, \ \forall p, p' \in \hat{P}, \ p \neq p', \ \forall t, t' \in \hat{T}, \ \forall o \in \hat{O} \tag{3.14}$$

Constraints (3.13) and (3.14) enforce that the pressing phase of press machine p at cycle t and the pressing phase of press machine p' at cycle t', which are assigned in the same oven o ($X_{pto} = X_{p't'o} = 1$), cannot be operated at the same time. These constraints, which are either-or constraints, help avoid an overlap of tasks in an oven.

12. Constraint for setting the auxiliary variable C'_{pt}:

$$C_{pt} - M\left(1 - \sum_{i=1}^{I}\sum_{k=1}^{K}\sum_{l=1}^{L} x_{iklpt}\right) \leq C'_{pt}, \ \forall p \in \hat{P}, \forall t \in \hat{T} \tag{3.15}$$

$$C'_{pt} \leq C_{pt} + M\left(1 - \sum_{i=1}^{I}\sum_{k=1}^{K}\sum_{l=1}^{L} x_{iklpt}\right), \ \forall p \in \hat{P}, \forall t \in \hat{T} \tag{3.16}$$

$$C'_{pt} \leq M\left(\sum_{i=1}^{I}\sum_{k=1}^{K}\sum_{l=1}^{L} x_{iklpt}\right), \ \forall p \in \hat{P}, \forall t \in \hat{T} \tag{3.17}$$

Constraints (3.15) – (3.17) ensure that if a panel type, a SST size, and a layout are assigned in cycle t of press machine p $\left(\sum\limits_{i=1}^{I}\sum\limits_{k=1}^{K}\sum\limits_{l=1}^{L} x_{iklpt} = 1\right)$, then variable C'_{pt} is equal to C_{pt}. Otherwise it is equal to 0. Note that if $\sum\limits_{i=1}^{I}\sum\limits_{k=1}^{K}\sum\limits_{l=1}^{L} x_{iklpt} = 1$, then Constraints (3.15) and (3.16) imply that $C'_{pt} = C_{pt}$, and Constraint (3.17) becomes redundant. On the other hand, if $\sum\limits_{i=1}^{I}\sum\limits_{k=1}^{K}\sum\limits_{l=1}^{L} x_{iklpt} = 0$, then Constraints (3.15) and (3.16) are redundant, and Constraint (3.17) sets variable $C'_{pt} = 0$. The variable C'_{pt} will be used to determine the makespan in Constraint (3.18).

13. Constraint for determining the makespan:

$$C_{\max} \geq C'_{pt},\ \forall p \in \hat{P}, \forall t \in \hat{T} \tag{3.18}$$

Constraint (3.18) defines the makespan, which is the maximum completion time of the last cycle of press machines that indeed perform the pressing process.

14. Constraint of decision variables:

$$\left.\begin{array}{ll}
x_{iklpt} \in \{0,1\} & \forall i \in \hat{I}, \forall k \in \hat{K}, \forall l \in \hat{L}, \forall p \in \hat{P}, \forall t \in \hat{T} \\
X_{pto} \in \{0,1\} & \forall p \in \hat{P}, \forall t \in \hat{T}, \forall o \in \hat{O} \\
Y_{ptp't'o} \in \{0,1\} & \forall p, p' \in \hat{P},\ p \neq p',\ \forall t, t' \in \hat{T},\ \forall o \in \hat{O} \\
A_{pt} \geq 0 & \forall p \in \hat{P}, \forall t \in \hat{T} \\
B_{pto} \geq 0 & \forall p \in \hat{P}, \forall t \in \hat{T}, \forall o \in \hat{O} \\
D_{pto} \geq 0 & \forall p \in \hat{P}, \forall t \in \hat{T}, \forall o \in \hat{O} \\
C_{pt} \geq 0 & \forall p \in \hat{P}, \forall t \in \hat{T} \\
C'_{pt} \geq 0 & \forall p \in \hat{P}, \forall t \in \hat{T} \\
C_{\max} \geq 0 &
\end{array}\right\} \tag{3.19}$$

The size complexity of Model 1, which includes NBV, NCV, and NC, is shown in Table 3.1.

Table 3.1: The size complexity of Model 1.

Type	**Model 1**
Binary variable	$IKLPT + PTO + P(P-1)T^2O$
Continuous variable	$3PT + 2PTO + 1$
Constraint	$8PT + 2P(T-1) + 2P(P-1)T^2O + 3PTO + IKLPT + I$

The proposed Model 1 demonstrates an application of MILP to solve the pressing process scheduling. From the solution of Model 1, the proper panel type, the SST size, and the layout which could be assigned to each press machine cycle are provided. Furthermore, the solution gives the details that which press machine cycle will be sent into which oven for performing the pressing phase as well as its starting and end times. Model 1 can be an option to find an optimal schedule of the pressing process for any PCB manufacturer.

3.3 Improved MILP model (Model 2)

Although Model 1 in the previous section can be a MILP model for the pressing process scheduling, the size complexity of this model is still large due to a lot of binary variables and constraints. It uses a large number of binary variables and constraints to formulate the constraint for avoiding an overlap in an oven. Furthermore, the starting time of the pressing phase of a press machine cycle is defined using too many continuous variables in Model 1. In addition, it defines the completion time variables, which is not necessary to be defined in the model. These can cause poor performance to the MILP model. This section proposes the second MILP model, which is called Model 2, for scheduling the pressing process. The proposed Model 2 is an improvement of Model 1, where NBV, NCV, NC, and the dimensionality of some decision variables in the model are reduced. The notations used in Model 2 are as follows.

Indices:
Parameters:
Sets:

} The indices, parameters, and sets used in Model 2 are the same as in Model 1.

Decision variables:

x_{iklpt}	Binary variable which is equal to 1 if panel type i is assigned with SST size k and layout l to press machine p at cycle t (The same as in Model 1).
X_{pto}	Binary variable which is equal to 1 if press machine p at cycle t is sent into oven o for operating the pressing phase (The same as in Model 1).
$Y_{ptp't'}$	Binary variable which is equal to 1 if press machine p at cycle t precedes press machine p' at cycle t' in the same oven.
A_{pt}	Continuous variable representing the starting time of the lay-up phase of press machine p at cycle t (The same as in Model 1).
B_{pt}	Continuous variable representing the starting time of the pressing phase of press machine p at cycle t.

A'_{pt} The auxiliary continuous variable, which is equal to A_{pt} if a panel type, a SST size, and a layout are assigned in press machine p at cycle t. Otherwise, it is equal to 0.

$C_{\max}$ Continuous variable representing the makespan (The same as in Model 1).

The proposed Model 2 for scheduling the pressing process can be stated as follows.

$$\min \quad C_{\max} \tag{3.20}$$

$$\text{s.t.} \quad \sum_{i=1}^{I}\sum_{k=1}^{K}\sum_{l=1}^{L} x_{iklpt} \le 1 \qquad \forall p \in \hat{P}, \forall t \in \hat{T} \tag{3.21}$$

$$x_{iklpt} \le a_{ikl} \qquad \forall i \in \hat{I}, \forall k \in \hat{K}, \forall l \in \hat{L}, \forall p \in \hat{P}, \forall t \in \hat{T} \tag{3.22}$$

$$\sum_{k=1}^{K}\sum_{l=1}^{L}\sum_{p=1}^{P}\sum_{t=1}^{T} x_{iklpt}(m \cdot a_{ikl}) \ge d_i \qquad \forall i \in \hat{I} \tag{3.23}$$

$$\sum_{i=1}^{I}\sum_{k=1}^{K}\sum_{l=1}^{L} x_{iklp(t-1)} \ge \sum_{i=1}^{I}\sum_{k=1}^{K}\sum_{l=1}^{L} x_{iklpt} \qquad \forall p \in \hat{P}, \forall t \in \hat{T} - \{1\} \tag{3.24}$$

$$\sum_{o=1}^{O} X_{pto} = 1 \qquad \forall p \in \hat{P}, \forall t \in \hat{T} \tag{3.25}$$

$$A_{pt} \ge A_{p,t-1} + 3n \qquad \forall p \in \hat{P}, \forall t \in \hat{T} - \{1\} \tag{3.26}$$

$$B_{pt} = A_{pt} + n \qquad \forall p \in \hat{P}, \forall t \in \hat{T} \tag{3.27}$$

$$B_{p't'} \ge B_{pt} + n - (3 - Y_{ptp't'} - X_{pto} - X_{p't'o})M \qquad \forall p, p' \in \hat{P}, p < p', \forall t, t' \in \hat{T}, \forall o \in \hat{O} \tag{3.28}$$

$$B_{pt} \ge B_{p't'} + n - (2 + Y_{ptp't'} - X_{pto} - X_{p't'o})M \qquad \forall p, p' \in \hat{P}, p < p', \forall t, t' \in \hat{T}, \forall o \in \hat{O} \tag{3.29}$$

$$A_{pt} - M\left(1 - \sum_{i=1}^{I}\sum_{k=1}^{K}\sum_{l=1}^{L} x_{iklpt}\right) \le A'_{pt} \qquad \forall p \in \hat{P}, \forall t \in \hat{T} \tag{3.30}$$

$$A'_{pt} \le A_{pt} + M\left(1 - \sum_{i=1}^{I}\sum_{k=1}^{K}\sum_{l=1}^{L} x_{iklpt}\right) \qquad \forall p \in \hat{P}, \forall t \in \hat{T} \tag{3.31}$$

$$A'_{pt} \le M\left(\sum_{i=1}^{I}\sum_{k=1}^{K}\sum_{l=1}^{L} x_{iklpt}\right) \qquad \forall p \in \hat{P}, \forall t \in \hat{T} \tag{3.32}$$

$$C_{\max} \ge A'_{pt} + 3n \qquad \forall p \in \hat{P}, \forall t \in \hat{T} \tag{3.33}$$

and,

$$
\begin{aligned}
&x_{iklpt} \in \{0,1\} && \forall i \in \hat{I}, \forall k \in \hat{K}, \forall l \in \hat{L}, \forall p \in \hat{P}, \forall t \in \hat{T},\\
&X_{pto} \in \{0,1\} && \forall p \in \hat{P}, \forall t \in \hat{T}, \forall o \in \hat{O},\\
&Y_{ptp't'} \in \{0,1\} && \forall p, p' \in \hat{P},\ p < p',\ \forall t, t' \in \hat{T},\\
&A_{pt} \geq 0 && \forall p \in \hat{P}, \forall t \in \hat{T},\\
&B_{pt} \geq 0 && \forall p \in \hat{P}, \forall t \in \hat{T},\\
&A'_{pt} \geq 0 && \forall p \in \hat{P}, \forall t \in \hat{T},\\
&C_{\max} \geq 0.
\end{aligned}
$$

The objective function (3.20) and Constraints (3.21) – (3.25) in Model 2 are the same as the objective function (3.1) and Constraints (3.2) – (3.6) in Model 1, respectively.

Constraint (3.26) in Model 2 is equivalent to Constraint (3.8) in Model 1. Constraint (3.26) combines Constraints (3.10) and (3.8) by replacing variable C_{pt} with $A_{pt} + 3n$ (as indicated in Constraint (3.10)) in Constraint (3.8). It ensures that the starting time of a cycle of a press machine (A_{pt}) must be greater than or equal to the completion time of the previous cycle. Note that the starting time of the previous cycle ($A_{p,t-1}$) plus the processing time of one cycle ($3n$) is equal to the completion time of the previous cycle.

Constraint (3.27) in Model 2 is equivalent to Constraint (3.9) in Model 1. Constraint (3.27) represents that the starting time of the pressing phase of press machine p at cycle t (B_{pt}) is equal to the starting time of this cycle (A_{pt}) plus the processing time (n) required in the lay-up phase. Note that RHS of Constraints (3.27) and (3.9) are the same while LHS of both constraints also represent the same value, which can be explained as follows. From LHS of Constraint (3.9), there is only one non-zero value of variable $B_{pto'}$ (assume that press machine p at cycle t is assigned to oven o' or $X_{pto'} = 1$) because all variables B_{pto}, $\forall o \in \hat{O} - \{o'\}$ will be 0 from Constraint (3.7), while LHS of Constraint (3.27) or B_{pt} is the starting time of the pressing phase of press machine p at cycle t in its assigned oven or oven o' (from variable $X_{pto'}$). Hence, LHS of both constraints are also the same value, and it means that Constraint (3.27) is equivalent to Constraint (3.9).

The role of Constraints (3.28) and (3.29) in Model 2 are equivalent to the role of Constraints (3.13) and (3.14) in Model 1. These constraints assure that the pressing phase of press machine p at cycle t and the pressing phase of press machine p' at cycle t', which are assigned in the same oven, cannot be operated simultaneously. More details will be explained later.

Constraints (3.30) – (3.32) in Model 2 are similar to constraints (3.15) – (3.17) in Model 1. These constraints are used to enforce that if a panel type, a SST size, and a layout are assigned in cycle t of press machine p, then $A'_{pt} = A_{pt}$. Otherwise $A'_{pt} = 0$. The auxiliary variable A'_{pt} will be used to determine the value of the makespan in Constraint (3.33).

Constraint (3.33) in Model 2 determines the makespan, which is equivalent to constraint (3.18) in Model 1.

Compared with Model 1 in Section 3.2, Model 2 contains various improved parts as follows:

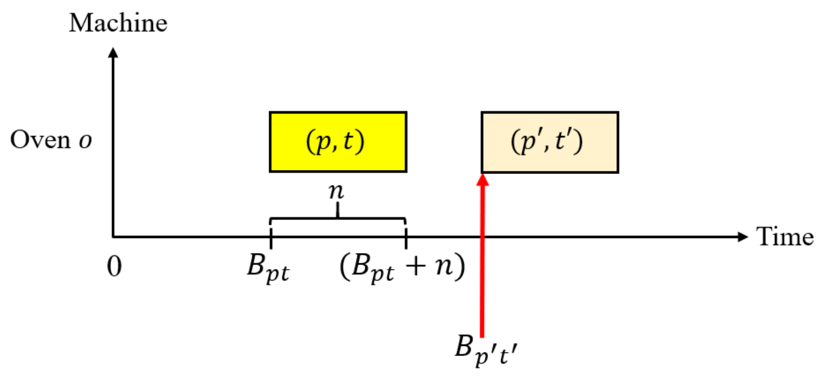
1. The new binary variable $Y_{ptp't'}$ in Model 2 replaces the old binary variable $Y_{ptp't'o}$. Note that both variables have the same role to prevent an overlap of any two tasks of the pressing phase that are assigned in the same oven. Nevertheless, the oven is not referenced in the definition of $Y_{ptp't'}$. Model 2 can still retrieve the information of the oven from variables X_{pto} and $X_{p't'o}$.

For example, suppose that variables $X_{111} = X_{221} = 1$. This means that the pressing phases in cycle 1 of press machine 1 and in cycle 2 of press machine 2 will be processed in the same oven 1. Assume that $Y_{11221} = 1$ in Model 1 and $Y_{1122} = 1$ in Model 2. The variable $Y_{11221} = 1$ in Model 1 means that the pressing phase in cycle 1 of press machine 1 is processed before the pressing phase in cycle 2 of press machine 2 in oven 1. On the other hand, variable $Y_{1122} = 1$ in Model 2 means that the pressing phase in cycle 1 of press machine 1 is processed before the pressing phase in cycle 2 of press machine 2 in the same oven. Even though the information of the oven is not provided by variable Y_{1122}, variables $X_{111} = X_{221} = 1$ still tells that the pressing phases in both cycle 1 of press machine 1 and cycle 2 of press machine 2 are performed in oven 1.

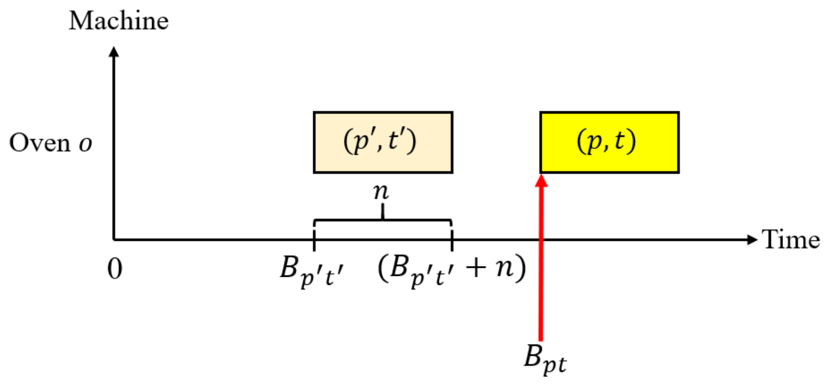
Due to the replacement of variable $Y_{ptp't'o}$ by $Y_{ptp't'}$, NBV in the model is reduced. It can also reduce the dimensionality of decision variables because variable $Y_{ptp't'o}$ has five indices, but variable $Y_{ptp't'}$ has four indices.

2. Model 2 uses NC less than Model 1 to formulate the constraint for avoiding an overlap in an oven. The non-overlapping constraints (3.28) and (3.29) in Model 2 are formulated only for cases $p < p'$, while the non-overlapping constraints (3.13) and (3.14) in Model 1 are formulated for cases $p \neq p'$. In fact, to prevent an overlap in an oven, the constraint only for cases $p < p'$ is required, which can be explained as follows. From Constraints (3.28) and (3.29) in Model 2, either one of them will be active for any two pressing phases of press machine p at cycle t and press machine p' at cycle t', which are performed in the same oven o ($X_{pto} = X_{p't'o} = 1$). If both tasks are not performed in the same oven o, both Constraints (3.28) and (3.29) will be redundant. If both tasks are performed in the same oven o and $Y_{ptp't'} = 1$, Constraint (3.28) is active (since $Y_{ptp't'} = X_{pto} = X_{p't'o} = 1$) while Constraint (3.29) is redundant. The activation of Constraint (3.28) means that the completion time of the pressing phase of press machine p at cycle t must be less than or equal to the starting time of the pressing phase of press machine p' at cycle t' in the same oven as shown in Figure 3.4(a). In other words, the task from press machine p at cycle t is operated before the task from press machine p' at cycle t'. On the other hand, if $Y_{ptp't'} = 0$, Constraint (3.29) is active (since $Y_{ptp't'} = 0$ and $X_{pto} = X_{p't'o} = 1$) while Constraint (3.28) is redundant. The activation of Constraint (3.29) means that the completion time of the pressing phase of press machine p' at cycle t' must be less than or equal to the starting time of the pressing phase of press machine p at cycle t in the same oven as shown in Figure 3.4(b). In other words, the task from press machine p at cycle t is processed after the task from press machine p' at cycle t'. These assure that any two pressing phases from any two press machine cycles assigned to the same oven cannot be done at the same time, according to the cases $p < p'$. Therefore, the non-overlapping condition in the oven is clearly defined, and the constraints in cases $p > p'$ is not necessary.

Note that, from Constraints (3.13) and (3.14) in Model 1, the constraint only in cases $p < p'$ are needed to prevent an overlap in an oven with the similar reason as explained for Constraints (3.28) and (3.29) in Model 2. This is because if $Y_{ptp't'o} = 1$,



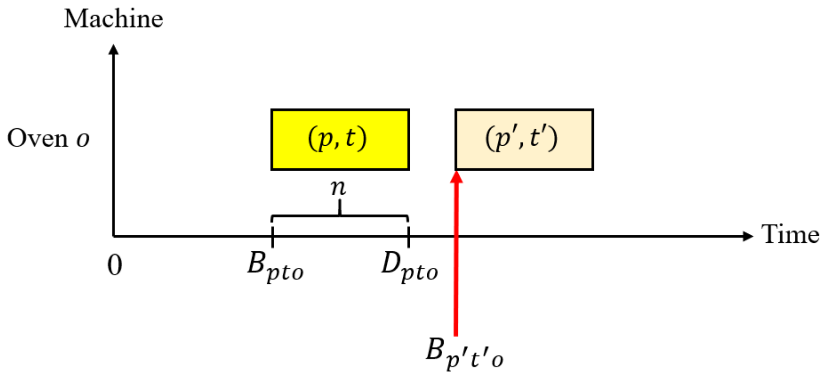
(a) Case $Y_{ptp't'} = 1$



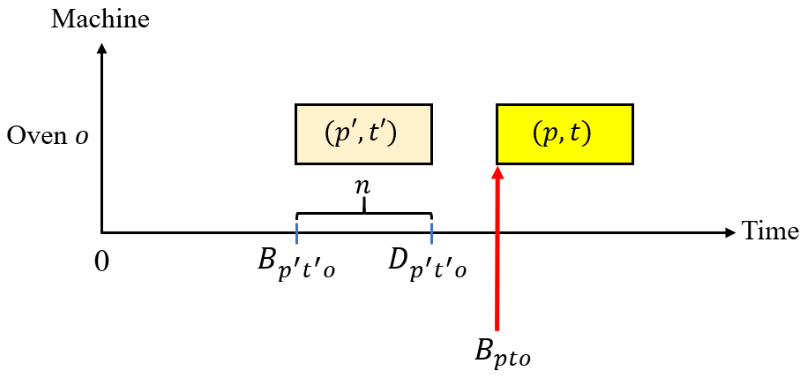
(b) Case $Y_{ptp't'} = 0$

Figure 3.4: Interpretation of Constraints (3.28) and (3.29).

Constraint (3.13) is redundant while Constraint (3.14) is active, which means that the starting time of the pressing phase of press machine p' at cycle t' (in oven o) must be greater than or equal to the completion time of the pressing phase of press machine p at cycle t (in oven o) as shown in Figure 3.5(a). On the other hand, if $Y_{ptp't'o} = 0$, Constraint (3.14) is redundant while Constraint (3.13) is active, which means that the starting time of the pressing phase of press machine p at cycle t (in oven o) must be greater than or equal to the completion time of the pressing phase of press machine p' at cycle t' (in oven o) as shown in Figure 3.5(b). According to the cases $p < p'$, it assures that any two tasks in the same oven cannot overlap. Thus, the constraint in cases $p > p'$ is not necessary. In fact, the constraint in cases $p > p'$ have the same role or are equivalent to the constraint in cases $p < p'$, i.e., the constraint in cases $p > p'$ are redundant. Therefore, Model 1 have the redundant constraints while Model 2 eliminates these redundant constraints from the model. NC for avoiding an overlap of tasks in an oven in Model 2 is halved comparing



(a) Case $Y_{ptp't'o} = 1$



(b) Case $Y_{ptp't'o} = 0$

Figure 3.5: Interpretation of Constraints (3.13) and (3.14).

with Model 1.

3. The new continuous variable B_{pt} in Model 2, which is without reference to the oven, replaces the old continuous variable B_{pto}. Both variables represent the starting time of the pressing phase of press machine p at cycle t in its assigned oven. For simplicity, assume that the pressing phase in cycle t of press machine p is processed in oven o'. In Model 1, Constraint (3.7), which controls variable B_{pto}, $\forall o \in \hat{O} - \{o'\}$ to be zero, is needed in order to have only the non-zero value of $B_{pto'}$ in the model. Nevertheless, this constraint is not necessary in Model 2 since the model uses variable B_{pt}, which does not reference the oven. Even though the information of the oven is not provided by variable B_{pt} for the task from press machine p at cycle t, it can be retrieved from variable X_{pto}.

For example, suppose that variable $X_{221} = 1$. This means that oven 1 will perform the pressing phase of press machine 2 at cycle 2. Assume that variable $B_{221} = 480$

minutes in Model 1, and variable $B_{22} = 480$ minutes in Model 2. The former means that the pressing phase in cycle 2 of press machine 2 is started at time 480 in oven 1, while the latter means that the pressing phase in cycle 2 of press machine 2 is started at time 480. Even though the information of the oven is not provided by variable B_{22}, variable $X_{221} = 1$ still tells that the pressing phase in cycle 2 of press machine 2 is performed in oven 1.

Due to the replacement of variable B_{pto} by B_{pt}, NCV in the model is reduced. It can also reduce the dimensionality of decision variable because variable B_{pto} has three indices, but variable B_{pt} has only two indices. Furthermore, NC is also reduced because it can eliminate Constraint (3.7) from the model.

4. The completion time variables are not used in Model 2, which include the completion time of press machine p at cycle t (C_{pt}) and the completion time of the pressing phase of press machine p at cycle t in oven o (D_{pto}). This is because these values can be retrieved after the model is solved, where the former is equal to $A_{pt} + 3n$ and the latter is equal to $B_{pt}+n$. This can reduce NCV in the model. In addition, NC is also reduced because it can eliminate some constraints that contain the completion times, which include Constraint (3.7) and Constraints (3.10) – (3.12).

Table 3.2: The size complexity of Models 1 and 2.

Type	Model 1	Model 2
Binary variable	$IKLPT + PTO + P(P-1)T^2O$	$IKLPT + PTO + \frac{P(P-1)T^2}{2}$
Continuous variable	$3PT + 2PTO + 1$	$3PT + 1$
Constraint	$8PT + 2P(T-1) + 2P(P-1)T^2O + 3PTO + IKLPT + I$	$7PT + 2P(T-1) + P(P-1)T^2O + IKLPT + I$

NBV, NCV, and NC that are used to formulate Model 2 are shown in Table 3.2. From Table 3.2, it could be concluded that Model 2 outperforms Model 1 in terms of the size complexity because NBV, NCV, NC that Model 2 generated are less than those in Model 1. Both Models 1 and 2 show applications of MILP to solve the pressing process scheduling and can be used to find an optimal schedule of the pressing process for any PCB manufacturer. The computational complexities of both models will be compared in

Chapter 4.

3.4 Proposed 3P-PCB-PH Algorithm

Since the pressing process scheduling is complicated, a MILP model can be unsuitable when the problem size is large. A heuristic algorithm, which is called a three-phase-PCB-pressing heuristic (3P-PCB-PH) algorithm, is proposed in this section for solving the pressing process scheduling. In this algorithm, the pressing process scheduling problem is solved in three phases. Phase 1 aims to match each panel type with an appropriate SST size as well as a layout and find the number of cycles required to satisfy the demands. In Phase 2, then, all cycles that must be used are scheduled to find the number of working cycles on each press machine, as well as their starting and end times. Lastly, each panel type is assigned to a working cycle of a press machine in Phase 3, together with its selected SST size and layout from Phase 1. The parameters in the 3P-PCB-PH algorithm are similar to those in Models 1 and 2. The 3P-PCB-PH algorithm has five steps, which are described below.

Step 1: Take the inputs $I, K, L, P, O, T, m, n, a_{ikl}$, $\forall i \in \hat{I}, k \in \hat{K}, l \in \hat{L}$, and $d_i, \forall i \in \hat{I}$.

Step 2 (Phase 1): Choosing a SST size and a layout for each panel type.

Algorithm 1 Phase 1 of the 3P-PCB-PH algorithm.

```
   Input: $I, K, L, m, d_i,\ \forall i \in \hat{I}$ and $a_{ikl},\ \forall i \in \hat{I}, k \in \hat{K}, l \in \hat{L}$
1: For each $i \in \hat{I}$, find $\bar{k}_i \in \hat{K}, \bar{l}_i \in \hat{L}$.
2: Compute $d_{c_i} = \left\lceil \frac{d_i}{m a_{i\bar{k}_i\bar{l}_i}} \right\rceil, \forall i \in \hat{I}$.
3: Compute $d_c = \sum_{i=1}^{I} d_{c_i}$.
   Output: $d_c,\ d_{c_i}, \forall i \in \hat{I}$ and $\bar{k}_i,\ \bar{l}_i, \forall i \in \hat{I}$
```

The inputs of Phase 1 consist of $I, K, L, m, d_i, \forall i \in \hat{I}$ and $a_{ikl}, \forall i \in \hat{I}, k \in \hat{K}, l \in \hat{L}$. In this phase, each panel type $i \in \hat{I}$ will be matched with a suitable SST size and a layout. A SST size $\bar{k}_i$ and a layout $\bar{l}_i$ which provide the maximum number of panels of type i per book are selected. From this selection, the number of panels of type i per book is $a_{i\bar{k}_i\bar{l}_i}$, and the number of panels of type i that can be obtained per cycle is $ma_{i\bar{k}_i\bar{l}_i}$. Let d_{c_i} be the minimum number of cycles required for pressing each panel type i. This value can be calculated from $d_{c_i} = \left\lceil \frac{d_i}{ma_{i\bar{k}_i\bar{l}_i}} \right\rceil$, where the expression $\lceil x \rceil$ is the smallest integer that is not smaller than x. The minimum number of total cycles required to satisfy the demands of all types of panels can be calculated by $d_c = \sum_{i=1}^{I} d_{c_i}$. Due to the assumption that the demands of panels from customers have a feasible schedule, the value d_c is not greater than the number of all available cycles $P \times T$. The algorithm for Phase 1 of 3P-PCB-PH is shown in Algorithm 1. The time complexity of Phase 1 algorithm is $O(IKL)$.

Step 3 (Phase 2): Scheduling the press machines and ovens.

To produce a schedule that minimizes the makespan, all d_c cycles are distributed to all press machines in this phase. The components of Phase 2 consist of the following.

1. Starting time matrix

The starting time matrix ($A = [A_{pt}]_{P\times T}$) stores the starting time of the lay-up phase of press machine p at cycle t (the starting time of press machine p at cycle t). In the beginning, A is initialized to be $[0]_{P\times T}$.

2. Completion time matrix

The completion time matrix stores the completion time of press machine p at cycle t. In the beginning, C is initialized to be $[0]_{P\times T}$.

3. Candidate list

The candidate list (Can) represents the next earliest available cycle number to use each press machine. In the beginning, Can is initialized to be $[1]_{1\times P}$ because the cycle that is available to start for each press machine is cycle 1.

4. Scheduled pressing job

The starting and end times of the pressing phase of a press machine cycle are collected in a scheduled pressing job, which is notated by (*start_time*, *end_time*, *press_machine*, *cycle*). The elements in a scheduled pressing job tuple represent the starting time, the completion time, the press machine number, and the cycle number, respectively. Assume that we have a scheduled pressing job $(240, 360, 1, 1)$, for example, it tells the information that this pressing job is processed from time 240 to 360 minutes and is the task of press machine 1 at cycle 1.

5. Oven schedule list

The oven schedule list ($Oven\_Schedule\_List$) stores the schedule pressing jobs to use in each oven in a sequential order. Each element in $Oven\_Schedule\_List$ is also a list that contains the schedule pressing job tuples assigned in the corresponding oven. An example of $Oven\_Schedule\_List$ is shown in Figure 3.6, where $O = 3$ and $n = 120$. From the first list in $Oven\_Schedule\_List$, two pressing jobs have already been assigned to oven 1. The first pressing job $(120, 240, 1, 1)$ means that oven 1 has to perform the pressing phase of press machine 1 at cycle 1 from 120 to 240 minutes, while the second pressing job $(480, 600, 1, 2)$ means that oven 1 has to perform the pressing phase of press machine 1 at cycle 2 from 480 to 600 minutes. Likewise, the list for oven 2 contains only one pressing job $(120, 240, 2, 1)$ that is already assigned. The third list (the list for oven 3) is empty, which means that there is currently no job assigned to oven 3 at this time. In the beginning, $Oven\_Schedule\_List$ is initialized to be the list of O empty lists $[[\ ]]_{1\times O}$. Later, the Phase 2 algorithm will append it with appropriate jobs.

$$Oven\_Schedule\_List = [\underbrace{[(120, 240, 1, 1), (480, 600, 1, 2)]}_{\text{Jobs in oven 1}}, \underbrace{[(120, 240, 2, 1)]}_{\text{Jobs in oven 2}}, \underbrace{[\quad]}_{\text{Jobs in oven 3}}]$$

Figure 3.6: An *Oven_Schedule_List* example.

6. Oven idle time interval list

The oven idle time interval list ($Oven\_Idle\_Time\_List$) stores idle time intervals of each oven in a consecutive order. Each element in $Oven\_Idle\_Time\_List$ is also a list that contains all idle time intervals in the corresponding oven. Figure 3.7 shows an example of $Oven\_Idle\_Time\_List$, which is corresponding to $Oven\_Schedule\_List$ in Figure 3.6. Note that, in the beginning, there is only one idle time interval $[0, \infty)$ in each oven because there is no task that had been assigned to it yet.

Algorithm 2 Phase 2 of the 3P-PCB-PH algorithm.

```
    Input: $P, O, T, n, d_c$
 1: Set $A = [0]_{P \times T},\ C = [0]_{P \times T},\ Can = [1]_{1 \times P}$,
        $Oven\_Schedule\_List = [[\ ]]_{1 \times O}$,
        $Oven\_Idle\_Time\_List = [[\,[0, \infty)\,]]_{1 \times O}$
 2: for $j = 1$ to $d_c$ do
 3:   Find $p'$, which is the press machine that has the minimum workload.
 4:   if $Can[p'] == 1$ then
 5:     Set $start\_time\_press\_machine = 0$
 6:   else
 7:     Set $start\_time\_press\_machine = C[p'][Can[p'] - 1]$
 8:   end if
 9:   Find $o'$, which is the oven that has the minimum workload.
10:   if Total processing time of oven $o' == 0$ then
11:     Set $A[p'][Can[p']] = start\_time\_press\_machine$,
            $C[p'][Can[p']] = start\_time\_press\_machine + 3n$,
            $start\_time\_oven = start\_time\_press\_machine + n$,
            $end\_time\_oven = start\_time\_oven + n$
12:     Add $(start\_time\_oven, end\_time\_oven, p', Can[p'])$ to $Oven\_Schedule\_List[o']$
13:     Update $Oven\_Idle\_Time\_List[o']$
14:   else
15:     Examine each idle time interval in $Oven\_Idle\_Time\_List[o']$ from left to
        right to find $start\_time\_oven$ and $end\_time\_oven$ for press machine $p'$ at
        cycle $Can[p']$ in oven $o'$.
16:     Set $A[p'][Can[p']] = start\_time\_oven - n$,
            $C[p'][Can[p']] = end\_time\_oven + n$
17:     Add $(start\_time\_oven, end\_time\_oven, p', Can[p'])$ to $Oven\_Schedule\_List[o']$
18:     Update $Oven\_Idle\_Time\_List[o']$
19:   end if
20:   Set $Can[p'] = Can[p'] + 1$
21: end for
    Output: $A$, $C$, and $Oven\_Schedule\_List$
```

$$Oven\_Idle\_Time\_List \;=\; \Big[\underbrace{[[0,120],[240,480],[600,\infty)]}_{\text{Idle time intervals in oven 1}},\ \underbrace{[[0,120],[240,\infty)]}_{\text{Idle time intervals in oven 2}},\ \underbrace{[[0,\infty)]}_{\text{Idle time intervals in oven 3}}\Big]$$

Figure 3.7: An *Oven_Idle_Time_List* example.

Following the introduction of all components, the Phase 2 algorithm is presented as follows. The inputs for the algorithm include P, O, T, n, and d_c. The value d_c is the total number of iterations of this algorithm. In each iteration, a press machine that has the minimum workload is chosen, say press machine p'. Then, the algorithm checks to see if the next earliest available cycle of this press machine ($Can[p']$) is the first cycle.

- If yes, set the starting time in cycle $Can[p']$ of press machine p' to 0.
- Otherwise, set the starting time in cycle $Can[p']$ of press machine p' to the completion time of the previous cycle.

Let *start_time_press_machine* be the starting time of press machine p' at cycle $Can[p']$. Note that this value does not yet represent the final starting time for this press machine cycle because we must first examine the feasibility with its assigned oven. To do the pressing phase, the cycle $Can[p']$ of press machine p' will then be assigned to the oven that has the minimum workload, say oven o'. After that, we check to see if oven o' has been used yet.

- If not, $Oven\_Idle\_Time\_List[o']$ contains only one idle time interval $[0,\infty)$. The press machine p' at cycle $Can[p']$ can begin the lay-up phase at *start_time_press_machine*, and it is put into oven o' sequentially to begin the pressing phase at $start\_time\_press\_machine + n$.
- If yes, all idle time intervals in $Oven\_Idle\_Time\_List[o']$ from left to right will be checked to find the earliest time that cycle $Can[p']$ of press machine p' can begin the pressing phase in oven o'. Figure 3.8 illustrates an example. Assume that oven

o' is oven 1 which was previously assigned the pressing phase of press machine 1 at cycle 1, and suppose that $n = 120$ minutes. Assume that p' is press machine 2, and $Can[2]$ is cycle 1. We have $start\_time\_press\_machine = 0$ because this is the first cycle. However, all idle time intervals in $Oven\_Idle\_Time\_List[1]$ will be examined from left to right because oven 1 has been used. From Figure 3.8, $Oven\_Idle\_Time\_List[1] = [[0, 120], [240, \infty)]$. The first idle time interval $[0, 120]$ is obviously infeasible to begin the pressing phase of this press machine cycle because the lay-up phase has not yet been completed. Thus, cycle 1 of press machine 2 can begin the pressing phase in oven 1 as quickly as possible at time 240 minutes in the second idle time interval $[240, \infty)$. For simplicity, let this time be $start\_time\_oven$. Consequently, the finishing time of the pressing phase (end_time_oven), the real starting time, and the real completion time of press machine p at cycle $Can[p']$ can be obtained as follows:

- ◇ $end\_time\_oven = start\_time\_oven + n$,
- ◇ $A[p'][Can[p']] = start\_time\_oven - n$,
- ◇ $C[p'][Can[p']] = end\_time\_oven + n$.

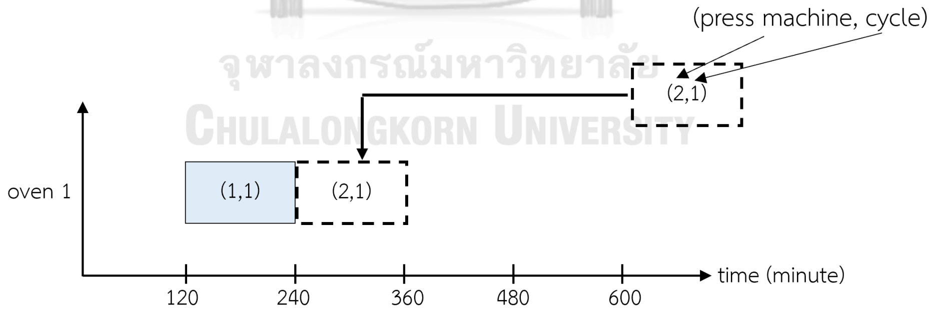


Figure 3.8: An example of finding $start\_time\_oven$.

After finding the $start\_time\_oven$, end_time_oven, $A[p'][Can[p']]$, and $C[p'][Can[p']]$, these values are updated in matrices A, C, $Oven\_Schedule\_List[o']$, and $Oven\_Idle\_Time\_List[o']$. Also, the value $Can[p']$ is increased by one in order to set the next cycle of press machine p' to be a new candidate. The algorithm will repeat until all d_c cycles are scheduled. The Phase 2 algorithm of 3P-PCB-PH is shown in Algorithm 2. The time complexity of the algorithm for Phase 2 is $O(P^2T^2)$.

Step 4 (Phase 3): Assigning the panel-SST-layout combinations to press machine working cycles.

The input for Phase 3 includes I, $d_{c_i}, \forall i \in \hat{I}$, and $\bar{k}_i$, $\bar{l}_i, \forall i \in \hat{I}$. Recall that Phase 2 provides the number of working cycles of each press machine. In Phase 3, each panel type is assigned to a working cycle of a press machine, together with its selected SST size and layout from Phase 1 as follows:

- For the first panel type, the d_{c_1} cycles are assigned to the first cycles of all press machines such that the work is equally distributed among the press machines.
- For the second panel type, the d_{c_2} cycles are assigned to the next available cycles of all the press machines in order that the work is equally distributed, and so on.

Algorithm 3 Phase 3 of the 3P-PCB-PH algorithm.

```
   Input: $I$, $d_{c_i}, \forall i \in \hat{I}$, and $\bar{k}_i$, $\bar{l}_i, \forall i \in \hat{I}$.
1: for $i = 1$ to $I$ do
2:     Distribute $d_{c_i}$ cycles for panel type $i$ (with SST size $\bar{k}_i$ and layout $\bar{l}_i$) to the
       earliest available cycles of all press machines equally as possible.
3: end for
   Output: The value $x_{iklpt}$, which is equal to 1.
```

The algorithm for Phase 3 of 3P-PCB-PH is shown in Algorithm 3. The output from this panel-cycle assignment can be used to interpret variable x_{iklpt}. Note that, from this assignment, the same-type panels are finished in a group, which is preferred in the real-world situation. An example of the result of this panel-cycle assignment will be shown in the next chapter. The time complexity of the Phase 3 algorithm is $O(PT)$.

Step 5: Interpreting the outputs.

After solving the problem using the proposed algorithm, we can get the outputs as follows.

1. Total number of finished goods of each panel type $i \in \hat{I}$: This value is equal to $(ma_{i\bar{k}_i\bar{l}_i})d_{c_i}$.
2. Schedule of press machines: The Gantt chart of press machines can be created from matrices A and C.
3. Schedule of ovens: The Gantt chart of ovens can be created from $Oven\_Schedule\_List$.
4. Variable x_{iklpt}: Variable $x_{iklpt}, \forall i \in \hat{I}, \forall k \in \hat{K}, \forall l \in \hat{L}, \forall p \in \hat{P}, \forall t \in \hat{T}$, which is equal to 1, can be interpreted from Phase 3.
5. Makespan: The makespan ($C_{\max}$) is the maximum element in matrix C.

From the 3P-PCB-PH algorithm, the time complexity includes $O(IKL)$ from Phase 1, $O(P^2T^2)$ from Phase 2, and $O(PT)$ from Phase 3. Therefore, the total time complexity of the 3P-PCB-PH algorithm is $O(P^2T^2 + IKL)$.

It is worth noting that PCB manufacturers prefer to complete each type of PCB in a group because it is convenient to provide materials and arrange the next work. The proposed Models 1 and 2 in Sections 3.2 and 3.3, respectively, can find an optimal schedule for a pressing process, but cycles of panels of the same type may not be scheduled continually. The proposed MILP models have this limitation while the proposed 3P-PCB-PH algorithm can handle this preference. Hence, the 3P-PCB-PH algorithm is more practicable to the real-world PCB production.

CHAPTER IV

EXPERIMENTS AND RESULTS

In this chapter, numerical experiments were conducted to evaluate the performance of all proposed approaches. We acquired real-world data from a PCB company and used it to generate the test problems. All problems were solved using Models 1 and 2 as well as the 3P-PCB-PH algorithm, and their results were compared. Section 4.1 shows the data and test problems. The computational results from Models 1 and 2 as well as the 3P-PCB-PH algorithm are shown in Sections 4.2 – 4.4, respectively. Lastly, the discussions are drawn in Section 4.5.

4.1 Data and test problems

The data obtained from a PCB company included the number of panel types (I), SST sizes (K), layouts (L), press machines (P), openings of a press machine (m), and ovens (O) as shown in Table 4.1. The company considered the planing horizon of 3 days. We assumed that the processing time of each phase in the pressing process (n) was 120 minutes, and the maximum number of available cycles of each press machine (T) was 12 as shown in Table 4.1. This is because a press machine cycle takes 360 minutes (6 hours). In three days, a press machine can conduct up to 12 cycles if it works continually.

Table 4.1: The data for generating the test problems.

Data type	**Parameter**	**Value**
Number of panel types	I	7
Number of SST sizes	K	6
Number of layouts	L	8
Number of press machines	P	6
Number of openings	m	10
Number of ovens	O	3
Processing time of each phase in the pressing process	n	120 min
Maximum number of available cycles of a press machine	T	12

Table 4.2 shows the information of each panel type, which consisted of the length or warp (a), width or fill (b), outer gap (G), and inner gap (g). Table 4.3 shows the sizes of each SST, which consisted of the warp (X) and fill (Y).

Table 4.2: Sizes and gaps of each panel type.

Panel	**Warp (a)**	**Fill (b)**	**Outer gap (G)**	**Inner gap (g)**
1	20.5	24	0.25	0.5
2	25.65	22.25	0.5	1
3	26	24	0.25	0.5
4	26.5	22.5	0.5	1
5	19	22.25	0.25	0.5
6	15	23.8	0.25	0.5
7	27.75	20.5	0.25	0.5

Table 4.3: Sizes of each SST.

SST size	**Warp (X)**	**Fill (Y)**
1	50	44
2	50	53
3	50	56
4	50	58
5	43	25.5
6	43	27

The illustration of eight layouts are shown in Figure 4.1. Note that only examples of the direction of an arrangement of panels on a SST are shown in this figure. The number of panels in the book is not limited to those depicted in this figure. For instance, the layout with two horizontal sections is illustrated in Figure 4.1(a), where the panels are positioned vertically in each section. The formulas for calculating the number of panels of type $i \in \hat{I}$ per book when panel type i is used with SST size $k \in \hat{K}$ and layout $l \in \hat{L}$ (a_{ikl}) of the PCB company are shown in Table 4.4. Note that the expression $\lfloor x \rfloor$ is the largest integer that is not greater than x.

In addition, for a variety of the problem sizes, this study also considered the planning horizons of 1.5, 2, and 4 days, with the maximum number of available cycles of each press

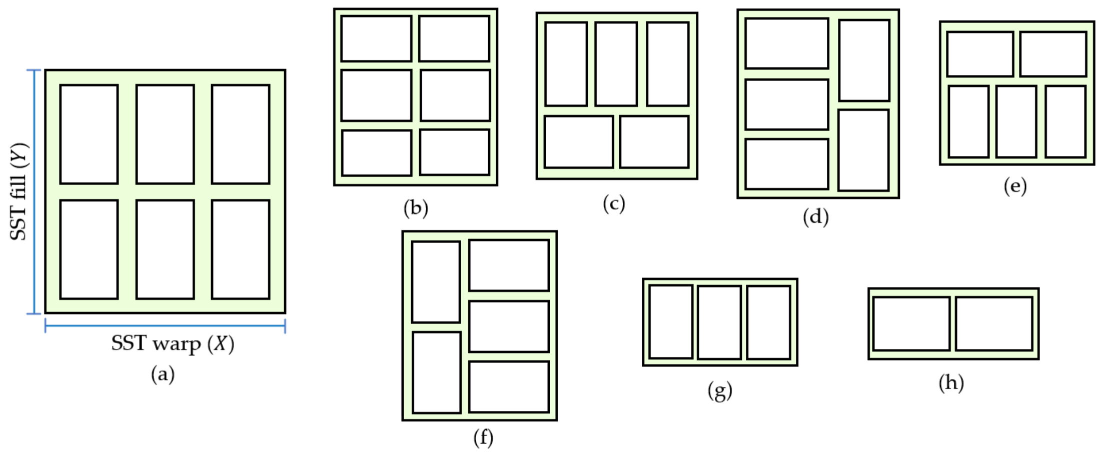


Figure 4.1: Illustration of eight layouts.

Table 4.4: Formulas for finding the number of panels of type i per book using SST size k and layout l.

Layout	$\boldsymbol{a_{ikl}}$
1	$\left\lfloor \frac{X-2(G-\frac{g}{2})}{a+g} \right\rfloor \times \left\lfloor \frac{Y-2(G-\frac{g}{2})}{b+g} \right\rfloor$
2	$\left\lfloor \frac{X-2(G-\frac{g}{2})}{b+g} \right\rfloor \times \left\lfloor \frac{Y-2(G-\frac{g}{2})}{a+g} \right\rfloor$
3	$\left\lfloor \frac{X-2(G-\frac{g}{2})}{a+g} \right\rfloor + \left(\left\lfloor \frac{X-2(G-\frac{g}{2})}{b+g} \right\rfloor \times \left\lfloor \frac{Y-b-G-2(G-\frac{g}{2})}{a+g} \right\rfloor \right)$
4	$\left\lfloor \frac{Y-2(G-\frac{g}{2})}{a+g} \right\rfloor + \left(\left\lfloor \frac{Y-2(G-\frac{g}{2})}{b+g} \right\rfloor \times \left\lfloor \frac{X-b-G-2(G-\frac{g}{2})}{a+g} \right\rfloor \right)$
5	$\left\lfloor \frac{X-2(G-\frac{g}{2})}{b+g} \right\rfloor + \left(\left\lfloor \frac{X-2(G-\frac{g}{2})}{a+g} \right\rfloor \times \left\lfloor \frac{Y-a-G-2(G-\frac{g}{2})}{b+g} \right\rfloor \right)$
6	$\left\lfloor \frac{Y-2(G-\frac{g}{2})}{b+g} \right\rfloor + \left(\left\lfloor \frac{Y-2(G-\frac{g}{2})}{a+g} \right\rfloor \times \left\lfloor \frac{X-a-G-2(G-\frac{g}{2})}{b+g} \right\rfloor \right)$
7	$\left\lfloor \frac{X-2(G-\frac{g}{2})}{a+g} \right\rfloor$
8	$\left\lfloor \frac{X-2(G-\frac{g}{2})}{b+g} \right\rfloor$

machine of 6, 8, and 16, respectively. The test problems were generated using the acquired data. Furthermore, for different problem sizes, parameters P and O in some test problems differed slightly from the real data. In each problem, the demand of each panel type i (d_i)

was randomly generated. The test problems were classified into four groups, i.e., small, medium, large, and extra-large problem, according to NBV. The small, medium, large, and extra-large problems consisted of Problems S1 – S5, M1 – M8, L1 – L9, and E1 – E9, respectively, as shown in Tables 4.5 – 4.8. A problem that has three panel types consists of panel types 1 – 3 in Table 4.2. Likewise, a problem with four, five, six, or seven panel types corresponds to panel types 1 – 4, 1 – 5, 1 – 6, or 1 – 7 in Table 4.2, respectively.

Table 4.5: The small problems.

No.	$\boldsymbol{I}$	$\boldsymbol{K}$	$\boldsymbol{L}$	$\boldsymbol{P}$	$\boldsymbol{O}$	$\boldsymbol{T}$	$\boldsymbol{d_i, i \in \hat{I}}$
S1	3	6	8	3	2	6	110, 150, 125
S2	3	6	8	3	2	8	200, 220, 230
S3	3	6	8	3	2	12	270, 250, 210
S4	3	6	8	4	2	6	110, 150, 125
S5	3	6	8	4	3	6	110, 150, 125

Table 4.6: The medium problems.

No.	$\boldsymbol{I}$	$\boldsymbol{K}$	$\boldsymbol{L}$	$\boldsymbol{P}$	$\boldsymbol{O}$	$\boldsymbol{T}$	$\boldsymbol{d_i, i \in \hat{I}}$
M1	3	6	8	6	3	6	300, 300, 300
M2	3	6	8	6	3	8	450, 480, 500
M3	3	6	8	6	3	12	720, 900, 600
M4	4	6	8	6	3	6	200, 300, 400, 100
M5	4	6	8	6	3	8	300, 400, 200, 500
M6	4	6	8	6	3	12	500, 700, 700, 500
M7	5	6	8	6	3	6	200, 250, 200, 250, 200
M8	5	6	8	6	3	8	400, 300, 200, 250, 300

4.2 Computational results of Model 1

In this section, IBM ILOG CPLEX 12.6 software was used to solve all test problems using Model 1 on a personal computer with 8 GB RAM and a core i7 2.20 GHz CPU. The time limit for solving each problem was set as 2 hours.

Table 4.7: The large problems.

No.	I	K	L	P	O	T	$d_i, i \in \hat{I}$
L1	5	6	8	6	3	12	500, 500, 500, 500, 500
L2	5	6	8	7	3	12	500, 500, 500, 500, 500
L3	5	6	8	6	4	12	500, 500, 500, 500, 500
L4	6	6	8	6	3	12	500, 360, 220, 180, 380, 720
L5	6	6	8	7	3	12	500, 360, 220, 180, 380, 720
L6	6	6	8	6	4	12	500, 360, 220, 180, 380, 720
L7	7	6	8	6	3	12	300, 325, 290, 425, 450, 475, 200
L8	7	6	8	7	3	12	300, 325, 290, 425, 450, 475, 200
L9	7	6	8	6	4	12	300, 325, 290, 425, 450, 475, 200

Table 4.8: The extra-large problems.

No.	I	K	L	P	O	T	$d_i, i \in \hat{I}$
E1	5	6	8	6	3	16	660, 525, 740, 850, 480
E2	5	6	8	7	3	16	660, 525, 740, 850, 480
E3	5	6	8	6	4	16	660, 525, 740, 850, 480
E4	6	6	8	6	3	16	400, 495, 800, 630, 700, 800
E5	6	6	8	7	3	16	400, 495, 800, 630, 700, 800
E6	6	6	8	6	4	16	400, 495, 800, 630, 700, 800
E7	7	6	8	6	3	16	500, 420, 595, 375, 330, 680, 580
E8	7	6	8	7	3	16	500, 420, 595, 375, 330, 680, 580
E9	7	6	8	6	4	16	500, 420, 595, 375, 330, 680, 580

4.2.1 Computational results of Model 1 for the small problems

Table 4.9 shows the model size and computational results of Model 1 for the small problems. The model size consisted of NBV, NCV, and NC. The results consisted of the number of outputs or finished goods of each panel type, the makespan ($C_{\max}$), and CPU time.

As shown in Table 4.9, all small problems could be solved using Model 1 within a small computational time. NBV of each small problem is less than 8,500. In each problem, the optimal makespan was found, and the output of each panel type satisfied

Table 4.9: Computational results of Model 1 for the small problems.

No.	I	K	L	P	O	T	d_i	**Model Size**			**Results**		
								NBV	**NCV**	**NC**	**Outputs**	**CPU Time**	$C_{\max}$ **(min)**
S1	3	6	8	3	2	6	110, 150, 125	3,060	127	3,741	120, 160, 160	2.31 s	1,440*
S2	3	6	8	3	2	8	200, 220, 230	4,272	169	5,373	200, 240, 240	2.48 s	2,160*
S3	3	6	8	3	2	12	270, 250, 210	6,984	253	9,213	280, 280, 240	3.21 s	2,520*
S4	3	6	8	4	2	6	110, 150, 125	4,368	169	5,563	120, 160, 160	8.15 s	1,200*
S5	3	6	8	4	3	6	110, 150, 125	4,824	217	6,499	120, 160, 160	3.18 s	1,080*

*Optimal solution.

the demand. For example, Problem S1 had the optimal makespan of 1,440 minutes, and the number of finished goods of panel types 1 – 3 were 120, 160, 160, respectively. The maximum computational time for solving the small problems to get an optimal solution is only 8.15 seconds in Problem S4. Note that the demands of Problems S1, S4, and S5 are the same. According to the results of Problem S4, if the number of press machines of Problem S1 was increased by one, the makespan of Problem S1 could be reduced from 1,440 to 1,200 minutes (i.e., the pressing process of Problem S1 could be finished earlier for 240 minutes). However, if the number of press machines and ovens were increased by one from Problem S1, in accordance with Problem S5, the makespan of Problem S1 could be reduced from 1,440 to 1,080 minutes. These show that Model 1 can be used to determine which resources should be increased in order to reduce the production time.

4.2.2 Computational results of Model 1 for the medium problems

Table 4.10 shows the model size and computational results of Model 1 for the medium problems. NBV of each medium problem is between 8,500 to 30,000. The results show that Model 1 could solve all medium problems to an optimal solution. The computational times for solving the medium problems are still acceptable for a real-life application. The maximum computational time for solving the medium problems optimally is 9 minutes 31 seconds in Problem M7. Note that this value increased significantly when compared to the maximum computational time for solving the small problems that is only 8.15 seconds in Problem S4.

4.2.3 Computational results of Model 1 for the large problems

The model size and computational results of Model 1 for the large problems are shown in Table 4.11. NBV of each large problem is around 30,000 to 46,000. The results show that only Problems L1, L2, and L4 could be solved optimally using Model 1 within the 2-h time limit. For the other problems that could not be solved optimally, the best feasible solution that could be obtained within the time limit (incumbent solution) was reported in Table 4.11. When comparing the maximum computational time for solving large problems optimally (48 minutes 14 seconds in Problem L4) to that for solving the

Table 4.10: Computational results of Model 1 for the medium problems.

No.	***I***	***K***	***L***	***P***	***O***	***T***	d_i	**Model Size**			**Results**		
								NBV	**NCV**	**NC**	**Outputs**	**CPU Time**	$C_{\max}$ **(min)**
M1	3	6	8	6	3	6	300, 300, 300	8,532	325	12,339	320, 320, 320	32.81 s	1,560*
M2	3	6	8	6	3	8	450, 480, 500	12,816	433	19,335	480, 480, 520	16.79 s	2,520*
M3	3	6	8	6	3	12	720, 900, 600	23,544	649	37,647	720, 920, 600	1 m 28 s	3,600*
M4	4	6	8	6	3	6	200, 300, 400, 100	10,260	325	14,068	200, 320, 400, 120	20.65 s	1,800*
M5	4	6	8	6	3	8	300, 400, 200, 500	15,120	433	21,640	320, 400, 200, 520	3 m 59 s	2,280*
M6	4	6	8	6	3	12	500, 700, 700, 500	27,000	649	41,104	520, 720, 720, 520	4 m 2 s	3,960*
M7	5	6	8	6	3	6	200, 250, 200, 250, 200	11,988	325	15,797	200, 280, 200, 280, 200	9 m 31 s	1,920*
M8	5	6	8	6	3	8	400, 300, 200, 250, 300	17,424	433	23,945	400, 320, 200, 280, 320	49.31 s	2,520*

*Optimal solution.

Table 4.11: Computational results of Model 1 for the large problems.

No.	I	K	L	P	O	T	d_i	Model Size			Results		
								NBV	NCV	NC	Outputs	CPU Time	$C_{\max}$ (min)
L1	5	6	8	6	3	12	500, 500, 500, 500, 500	30,456	649	44,561	520, 520, 520, 520, 520	44 m 38 s	4,080*
L2	5	6	8	7	3	12	500, 500, 500, 500, 500	38,556	757	58,035	520, 520, 520, 520, 520	23 m 42 s	3,600*
L3	5	6	8	6	4	12	500, 500, 500, 500, 500	34,848	793	53,417	520, 520, 520, 520, 520	2 h	4,080^1
L4	6	6	8	6	3	12	500, 360, 220, 180, 380, 720	33,912	649	48,018	520, 360, 240, 200, 400, 770	48 m 14 s	3,360*
L5	6	6	8	7	3	12	500, 360, 220, 180, 380, 720	42,588	757	62,068	520, 360, 240, 200, 400, 770	2 h	3,000^1
L6	6	6	8	6	4	12	500, 360, 220, 180, 380, 720	38,304	793	56,874	520, 360, 240, 200, 400, 770	2 h	3,360^1
L7	7	6	8	6	3	12	300, 325, 290, 425, 450, 475, 200	37,368	649	51,475	320, 360, 320, 440 480, 490, 200	2 h	3,720^1
L8	7	6	8	7	3	12	300, 325, 290, 425, 450, 475, 200	46,620	757	66,101	320, 360, 320, 440 480, 490, 200	2 h	3,360^1
L9	7	6	8	6	4	12	300, 325, 290, 425, 450, 475, 200	41,760	793	60,331	320, 360, 320, 440 480, 490, 200	2 h	3,720^1

*Optimal solution. ^{1}The incumbent solution from Model 1.

Table 4.12: List of x_{iklpt} values which is equal to 1 in the optimal solution of Problem L1 using Model 1.

Press Machine	**Non-zero x_{iklpt}**
1	$x_{13111}, x_{51212}, x_{24213}, x_{51214}, x_{33215}, x_{51216}, x_{51217}, x_{24218}, x_{43219}, x_{3221,10}, x_{4321,11}$
2	$x_{14621}, x_{53622}, x_{53223}, x_{54524}, x_{32225}, x_{12626}, x_{43227}, x_{44228}, x_{32229}, x_{1222,10}, x_{3222,11}$
3	$x_{23431}, x_{51532}, x_{44233}, x_{43234}, x_{43235}, x_{43236}, x_{12437}, x_{34238}, x_{12239}, x_{2343,10}, x_{4423,11}$
4	$x_{14141}, x_{43242}, x_{32243}, x_{24244}, x_{13445}, x_{32246}, x_{32247}, x_{32248}, x_{24649}, x_{5414,10}, x_{5134,11}$
5	$x_{12151}, x_{14352}, x_{54553}, x_{23454}, x_{32255}, x_{23656}, x_{42257}, x_{32258}, x_{43259}, x_{2365,10}$
6	$x_{51361}, x_{13362}, x_{24663}, x_{42264}, x_{24665}, x_{23466}, x_{24467}, x_{54468}, x_{11269}, x_{1126,10}, x_{3426,11}$

Table 4.13: List of X_{pto} values which is equal to 1 in the optimal solution of Problem L1 using Model 1.

Press Machine	**Non-zero X_{pto}**
1	$X_{111}, X_{122}, X_{131}, X_{142}, X_{151}, X_{161}, X_{171}, X_{181}, X_{191}, X_{1,10,3}, X_{1,11,3}$
2	$X_{212}, X_{223}, X_{233}, X_{241}, X_{253}, X_{263}, X_{272}, X_{282}, X_{292}, X_{2,10,1}, X_{2,11,1}$
3	$X_{311}, X_{322}, X_{331}, X_{341}, X_{353}, X_{363}, X_{373}, X_{383}, X_{393}, X_{3,10,2}, X_{3,11,2}$
4	$X_{412}, X_{421}, X_{433}, X_{442}, X_{451}, X_{461}, X_{472}, X_{482}, X_{491}, X_{4,10,1}, X_{4,11,3}$
5	$X_{513}, X_{522}, X_{531}, X_{543}, X_{553}, X_{563}, X_{573}, X_{581}, X_{593}, X_{5,10,2}$
6	$X_{613}, X_{623}, X_{632}, X_{643}, X_{652}, X_{662}, X_{673}, X_{681}, X_{692}, X_{6,10,3}, X_{6,11,1}$

medium problems (9 minutes 31 seconds in Problem M7), it should be noted that the maximum computational time for solving large problems increased significantly.

The following is an example of an optimal solution from the proposed model. For the results of Problem L1, there were 520 outputs of each panel type, which satisfied the demand. Tables 4.12 and 4.13 show variables x_{iklpt} and X_{pto}, which were equal to 1, in the optimal solution of Problem L1, respectively. In addition, Figures 4.2 and 4.3 show the Gantt charts of press machines and ovens, respectively, from the optimal solution of Problem L1. Note that the same color in Figures 4.2 and 4.3 represents the same panel type.

From Table 4.12, the values x_{iklpt}, which were equal to 1, were sorted by the index

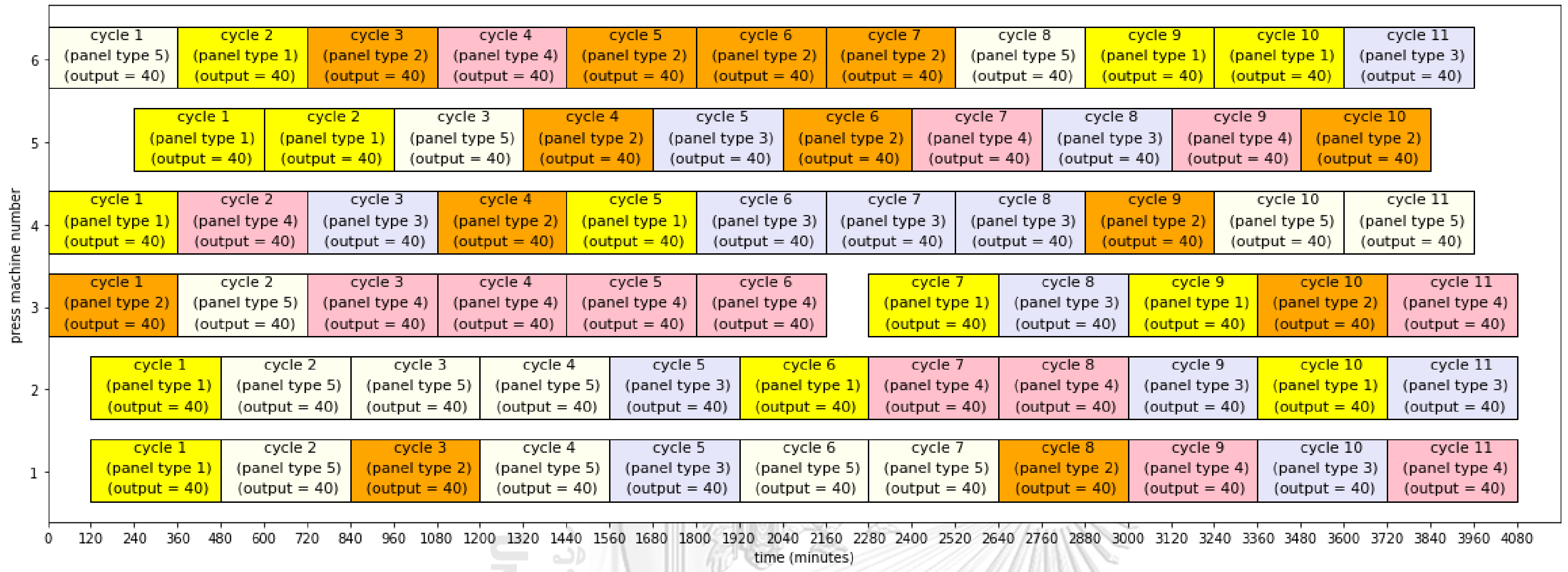


Figure 4.2: Gantt chart of press machines for Problem L1 using Model 1.

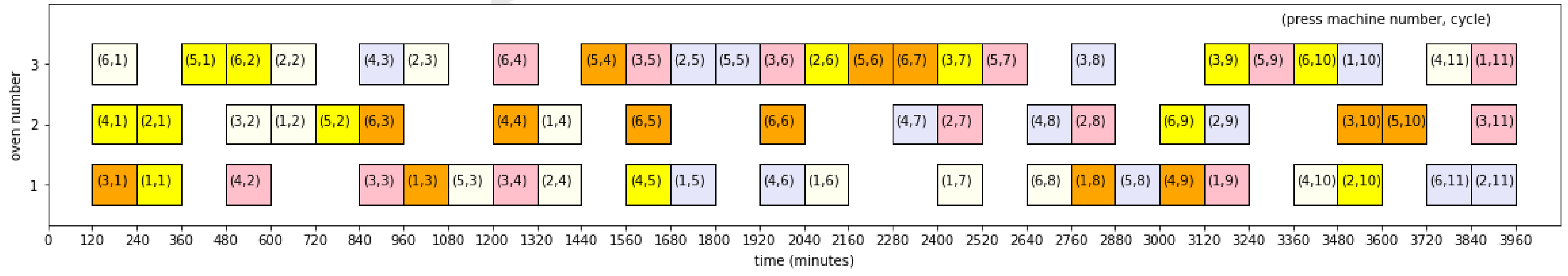


Figure 4.3: Gantt chart of ovens for Problem L1 using Model 1.

of cycle numbers in ascending order. For example, x_{13111} means that panel type 1, SST size 3, and layout 1 were used to create books for press machine 1 at cycle 1. Therefore, x_{13111} appears before x_{51212} in Table 4.12. In addition, the values X_{pto}, which were equal to 1, in Table 4.13 were sorted in a similar manner. For example, X_{212} means that the pressing phase of press machine 2 at cycle 1 was processed in oven 2. Therefore, X_{212} appears before X_{223} in Table 4.13.

Figure 4.2 shows the starting and completion times of each press machine cycle, while Figure 4.3 shows the time of the pressing phase for each press machine cycle. For instance, press machine 1 at cycle 1 operated the lay-up phase at 120 – 240 minutes (Figure 4.2), performed the pressing phase in oven 1 at 240 – 360 minutes (Figure 4.3), and cooled down at 360 – 480 minutes (Figure 4.2). The minimal makespan of this problem was 4,080 minutes (Figure 4.2).

4.2.4 Computational results of Model 1 for the extra-large problems

The model size and computational results of Model 1 for the extra-large problems are shown in Table 4.14. NBV of each extra-large problem is greater than 46,000. The results show that all extra-large problems could not be solved by Model 1 to reach optimality within the 2-h time limit. Therefore, Table 4.14 reports the incumbent solution that could be obtained within the time limit for each problem. When the problem size increased from small to extra large, it should be noted that NBV and NC of the test problems increased rapidly. A lot of binary variables and constraints in extra-large problems can cause a long computational time because NBV is the most important factor that influences the MILP model's performance, and NC is the next most important factor [17].

From numerical experiments, the results show that Model 1 is appropriate for solving small and medium problems, where NBV is less than 30,000. Model 1 could solve all small and medium problems using an acceptable time for a real-life application. However, only three large problems could be solved using Model 1, and the computational time to solve these problems optimally seem to be large when compared to the computational time for solving small and medium problems. The other problems in large problems and

Table 4.14: Computational results of Model 1 for the extra-large problems.

No.	I	K	L	P	O	T	d_i	Model Size			Results		
								NBV	NCV	NC	Outputs	CPU Time	$C_{\max}$ (min)
E1	5	6	8	6	3	16	660, 525, 740, 850, 480	46,368	865	70,937	680, 560, 760, 880, 480	2 h	5,280[1]
E2	5	6	8	7	3	16	660, 525, 740, 850, 480	59,472	1,009	93,511	680, 560, 760, 880, 480	2 h	4,560[1]
E3	5	6	8	6	4	16	660, 525, 740, 850, 480	54,144	1,057	86,585	680, 560, 760, 880, 480	2 h	5,160[1]
E4	6	6	8	6	3	16	400, 495, 800, 630, 700, 800	50,976	865	75,546	400, 520, 800, 640, 720, 840	2 h	5,520[1]
E5	6	6	8	7	3	16	400, 495, 800, 630, 700, 800	64,848	1,009	98,888	400, 520, 800, 640, 720, 840	2 h	4,800[1]
E6	6	6	8	6	4	16	400, 495, 800, 630, 700, 800	58,752	1,057	91,194	400, 520, 800, 640, 720, 840	2 h	5,520[1]
E7	7	6	8	6	3	16	500, 420, 595, 375, 330, 680, 580	55,584	865	80,115	520, 440, 600, 400 360, 700, 600	2 h	5,160[1]
E8	7	6	8	7	3	16	500, 420, 595, 375, 330, 680, 580	70,224	1,009	104,265	520, 440, 600, 400 360, 700, 600	2 h	4,560[1]
E9	7	6	8	6	4	16	500, 420, 595, 375, 330, 680, 580	63,360	1,057	95,803	520, 440, 600, 400 360, 700, 600	2 h	5,160[1]

[1]The incumbent solution from Model 1.

all extra-large problems could not be solved optimally by Model 1 within the 2-h time limit.

4.3 Computational results of Model 2

In this section, IBM ILOG CPLEX 12.6 software was used to solve all test problems using Model 2 on the same hardware environment as in the previous section. The time limit for solving each problem was also set as 2 hours. The results from Model 2 were compared with the results from Model 1 in the previous section.

The performance measures of size and computational complexities [10,13,16,18,19] were used to compare the performance between Models 1 and 2. The size complexity is measured by counting NBV, NCV, and NC, which are generated by the MILP model. If a MILP model produces fewer numbers of each size complexity factor than another model, it is more superior in terms of the size complexity. For the computational complexity, the criterion for deciding the performance between two MILP models consisted of the following.

1. If the first model could solve the problem to reach optimality within the time limit while the second model could not, the first model is clearly better than the second model in terms of the computational complexity.

2. If both models could solve the problem optimally within the time limit, the computational times for solving the problem using each model were compared. The lower the computational time value, the higher the model's performance.

3. If both models could not solve the problem to reach optimality within the time limit,

    - The incumbent solutions that could be obatined from each model were compared. Since the pressing process scheduling problem is a minimization problem, the smaller the incumbent makespan value, the higher the model's performance.

- The %gap from both models were compared if the incumbent solutions from both models have the same quality, i.e., the makespans from both incumbent solutions are the same value. The %gap is computed from $\left|\frac{\text{BestBound}-\text{BestInteger}}{\text{BestInteger}}\right|$ $\times 100\%$, which is the relative gap tolerance of the objective value for the solution from CPLEX. The BestBound is the current lower bound that the model could obtain within the time limit, while the BestInteger is the objective value of the incumbent solution that could be obtained from the model within the time limit. The %gap is equal to 0 when the problem could be solved to reach optimality. The lower the %gap value, the higher the model's performance.
- If the incumbent makespan and %gap from both models are the same, the computational times to reach the incumbent solution (time_{inc}) from each model were compared. The lower the time_{inc} value, the higher the model's performance.

4.3.1 Computational results of Model 2 for the small problems

The model size and computational results of Models 1 and 2 for the small problems are shown in Table 4.15. In addition, RPI_{time} was reported in the last column of Table 4.15. RPI_{time} is computed from $\frac{\text{CPUtime}_{\text{Model1}}-\text{CPUtime}_{\text{Model2}}}{\text{CPUtime}_{\text{Model1}}} \times 100\%$, which is the relative percentage improvement of the CPU times of Model 2 over Model 1. From Table 4.15, NBV, NCV, and NC of each small problem in Model 2 were less than those in Model 1. Figures 4.4 – 4.6 show NBV, NCV, and NC, respectively, that each model produced in each small problem. These show that Model 2 is evidently better than Model 1 in terms of the size complexity for the small problems. In each small problem, an optimal solution could be found by both Models 1 and 2, but Model 2 used smaller CPU time to find an optimal solution for each small problem. Figure 4.7 demonstrates the CPU times that Models 1 and 2 used for solving each small problem. This shows that Model 2 is also better than Model 1 in terms of the computational complexity for the small problems. The last column in Table 4.15 shows RPI_{time} of each small problem, and the last row of Table 4.15 shows that Model 2 had superior CPU time with average RPI_{time} of 36.84% for the small problems.

Table 4.15: Computational results of Models 1 and 2 for the small problems.

No.	Results of Model 1						Results of Model 2						$\mathbf{RPI_{time}}$
	Model Size			**Results**			**Model Size**			**Results**			
	NBV	**NCV**	**NC**	**Outputs**	**CPU Time**	$C_{\max}$ **(min)**	**NBV**	**NCV**	**NC**	**Outputs**	**CPU Time**	$C_{\max}$ **(min)**	
S1	3,060	127	3,741	120, 160, 160	2.31 s	1,440*	**2,736**	**55**	**3,183**	120, 160, 160	**1.51 s**	1,440*	34.63%
S2	4,272	169	5,373	200, 240, 240	2.48 s	2,160*	**3,696**	**73**	**4,437**	200, 240, 240	**1.76 s**	2,160*	29.03%
S3	6,984	253	9,213	280, 280, 240	3.21 s	2,520*	**5,688**	**109**	**7,233**	280, 280, 240	**1.82 s**	2,520*	43.30%
S4	4,368	169	5,563	120, 160, 160	8.15 s	1,200*	**3,720**	**73**	**4,531**	120, 160, 160	**5.29 s**	1,200*	35.09%
S5	4,824	217	6,499	120, 160, 160	3.18 s	1,080*	**3,744**	**73**	**4,963**	120, 160, 160	**1.84 s**	1,080*	42.14%
												Average	36.84%

*Optimal solution. The better values are in bold.

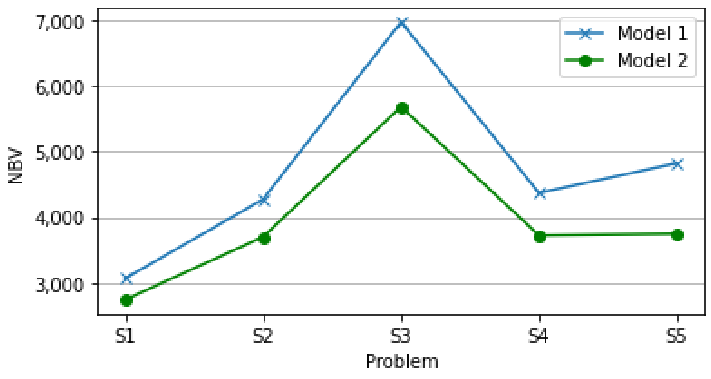


Figure 4.4: NBV comparison of Models 1 and 2 for the small problems.

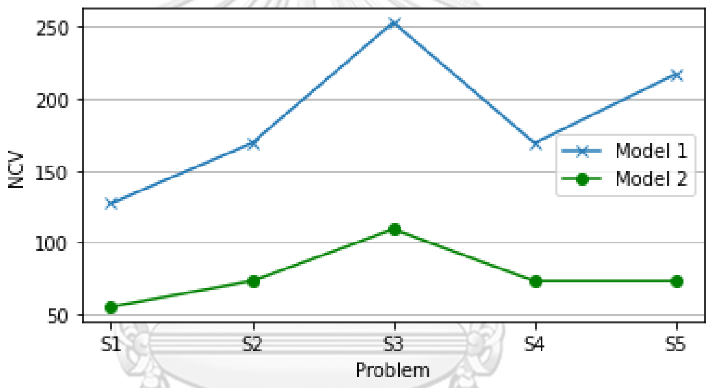


Figure 4.5: NCV comparison of Models 1 and 2 for the small problems.

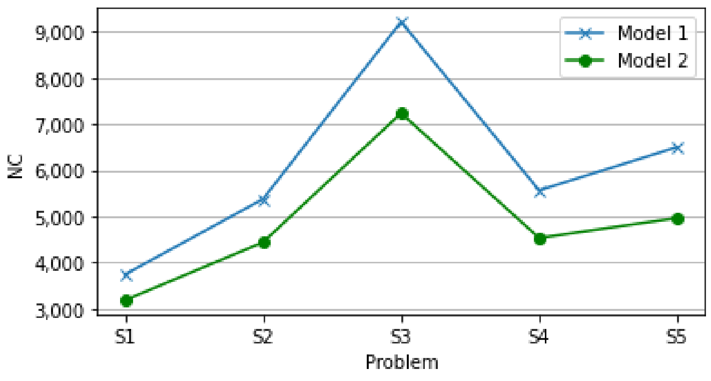


Figure 4.6: NC comparison of Models 1 and 2 for the small problems.

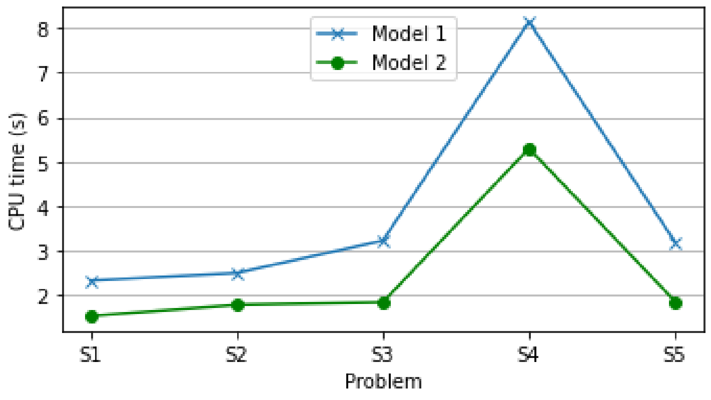


Figure 4.7: CPU time comparison of Models 1 and 2 for the small problems.

4.3.2 Computational results of Model 2 for the medium problems

Table 4.16 shows the model size and the computational results of Models 1 and 2 for the medium problems. For each medium problem, NBV, NCV, and NC that Model 2 generated were less than those that Model 1 generated. Figures 4.8 – 4.10 show NBV, NCV, and NC, respectively, that each model produced in each medium problem. This shows that, for the medium problems, Model 2 is better than Model 1 in terms of the size complexity. An optimal solution of each medium problem could be found by both Models 1 and 2, but Model 2 used smaller CPU time for solving each medium problem to an optimal solution. Figure 4.11 shows the CPU times that were used for solving each medium problem using Models 1 and 2. It shows that, for the medium problems, Model 2 is also better than Model 1 in terms of the computational complexity, and Model 2 had superior CPU time with average RPI_{time} of 69.98%.

4.3.3 Computational results of Model 2 for the large problems

Table 4.17 shows the model size and computational results of Models 1 and 2 for the large problems. The results consisted of the outputs, CPU time, makespan, and %gap. The column "CPU time" in Table 4.17 represents the computational time that CPLEX took to solve the model to reach optimality (%gap = 0), or 2 hours if CPLEX could not solve the model to reach optimality within the time limit (%gap $\neq$ 0). Note that it requires more computational time than 2 hours to solve the problem optimally

Table 4.16: Computational results of Models 1 and 2 for the medium problems.

No.	Results of Model 1						Results of Model 2						$\mathbf{RPI_{time}}$
	Model Size			Results			Model Size			Results			
	NBV	NCV	NC	Outputs	CPU Time	$C_{\max}$ (min)	NBV	NCV	NC	Outputs	CPU Time	$C_{\max}$ (min)	
M1	8,532	325	12,339	320, 320, 320	32.81 s	1,560*	**5,832**	**109**	**8,739**	320, 320, 320	**13.93 s**	1,560*	57.54%
M2	12,816	433	19,335	480, 480, 520	16.79 s	2,520*	**8,016**	**145**	**13,095**	480, 480, 520	**6.29 s**	2,520*	62.54%
M3	23,544	649	37,647	720, 920, 600	1 m 28 s	3,600*	**12,744**	**217**	**23,967**	720, 920, 600	**25.46 s**	3,600*	71.07%
M4	10,260	325	14,068	200, 320, 400, 120	20.65 s	1,800*	**7,560**	**109**	**10,468**	200, 320, 400, 120	**2.40 s**	1,800*	88.38%
M5	15,120	433	21,640	320, 400, 200, 520	3 m 59 s	2,280*	**10,320**	**145**	**15,400**	320, 400, 200, 520	**55.21 s**	2,280*	76.90%
M6	27,000	649	41,104	520, 720, 720, 520	4 m 2 s	3,960*	**16,200**	**217**	**27,424**	520, 720, 720, 520	**20.48 s**	3,960*	91.54%
M7	11,988	325	15,797	200, 280, 200, 280, 200	9 m 31 s	1,920*	**9,288**	**109**	**12,197**	200, 280, 200, 280, 200	**6 m 59 s**	1,920*	26.62%
M8	17,424	433	23,945	400, 320, 200, 280, 320	49.31 s	2,520*	**12,624**	**145**	**17,705**	400, 320, 200, 280, 320	**7.26 s**	2,520*	85.28%
												Average	69.98%

*Optimal solution. The better values are in bold.

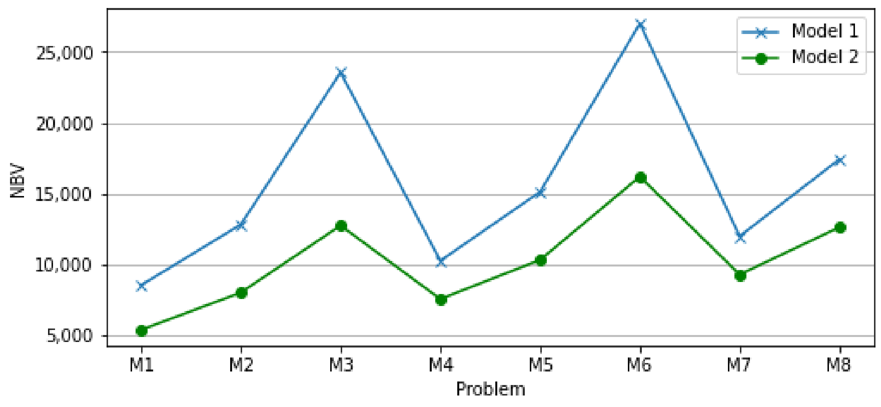


Figure 4.8: NBV comparison of Models 1 and 2 for the medium problems.

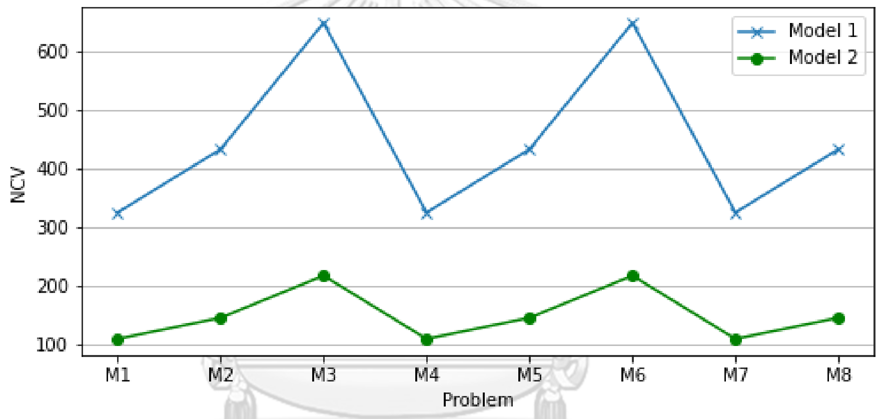


Figure 4.9: NCV comparison of Models 1 and 2 for the medium problems.

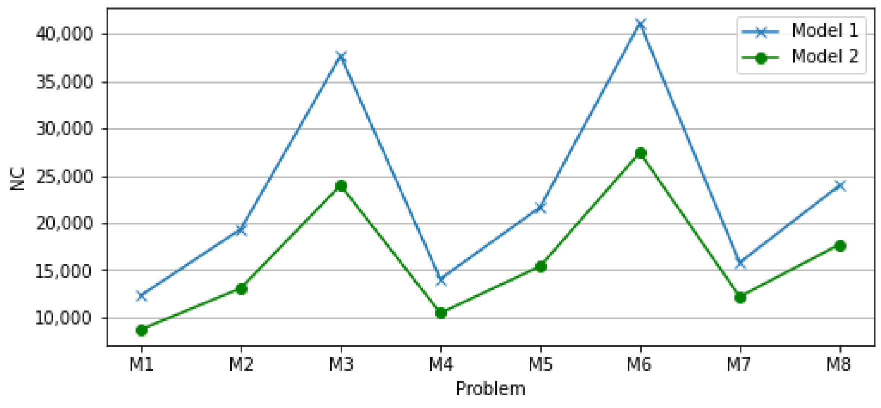


Figure 4.10: NC comparison of Models 1 and 2 for the medium problems.

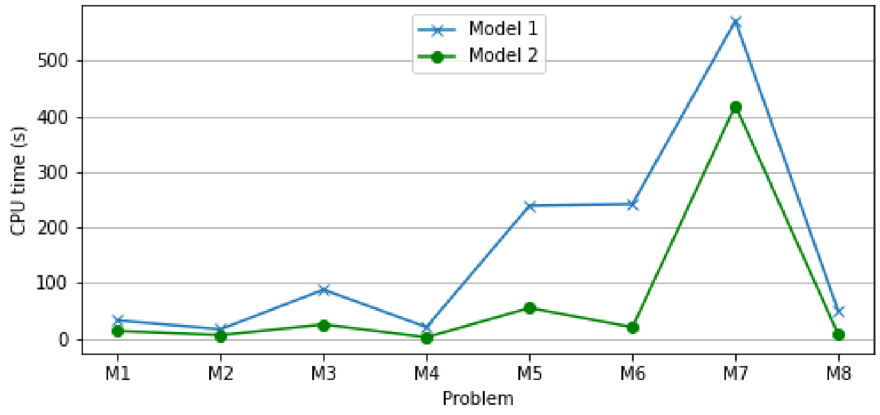


Figure 4.11: CPU time comparison of Models 1 and 2 for the medium problems.

for the problem with the solution that is obtained within the time limit and %gap $\neq$ 0. The column "$C_{\max}$" represents the optimal makespan or incumbent makespan that could be obtained within the time limit. The time$_{\text{inc}}$ is also reported in Table 4.17, which is the computational time to reach the incumbent solution. For the problem that could be solved to reach optimality, the time$_{\text{inc}}$ is the time that it first reached the optimal solution, but the optimality status at that time was not verified yet since the %gap $\neq$ 0 at that time. RPI$_{\text{time}}$ and RPI$_{\text{time}}^{\text{inc}}$ are also reported in the last column of Table 4.17, where RPI$_{\text{time}}^{\text{inc}}$ is computed from $\frac{(\text{time}_{\text{inc}})_{\text{Model1}} - (\text{time}_{\text{inc}})_{\text{Model2}}}{(\text{time}_{\text{inc}})_{\text{Model1}}} \times 100\%$, which represents the relative percentage improvement of time$_{\text{inc}}$ of Model 2 over Model 1.

From Table 4.17, NBV, NCV, and NC of each large problem in Model 2 were less than those in Model 1. Figures 4.12 – 4.14 show NBV, NCV, and NC, respectively, that each model produced in each large problem. It shows that, for the large problems, Model 2 is obviously better than Model 1 in terms of the size complexity.

Moreover, from Table 4.17, only three large problems (Problems L1, L2, and L4) could be solved optimally by Model 1 within the time limit. However, these problems could be solved optimally by Model 2 using smaller CPU time. Model 1 could not solve Problems L6, L7, and L9 to reach optimality within the time limit because %gap $\neq$ 0, while these problems could be solved by Model 2 to reach optimality within the time limit (%gap = 0%). For Problems L3, L5, and L8, both models could not solve them optimally, where the incumbent makespans from both models were the same, and the

Table 4.17: Computational results of Models 1 and 2 for the large problems.

No.	Results of Model 1								Results of Model 2								$\mathbf{RPI_{time}}$/ $\mathbf{RPI_{time}^{inc}}$
	Model Size			Results					Model Size			Results					
	NBV	**NCV**	**NC**	**Outputs**	**CPU Time**	**%gap**	$C_{\max}$ **(min)**	$\mathbf{Time_{inc}}$	**NBV**	**NCV**	**NC**	**Outputs**	**CPU Time**	**%gap**	$C_{\max}$ **(min)**	$\mathbf{Time_{inc}}$	
L1	30,456	649	44,561	520, 520, 520, 520, 520	44 m 38 s	0%	4,080*	4 m 44 s	**19,656**	**217**	**30,881**	520, 520, 520, 520, 520	**23 m 34 s**	0%	4,080*	**33.68 s**	47.20%/ 88.14%
L2	38,556	757	58,035	520, 520, 520, 520, 520	23 m 42 s	0%	3,600*	6 m 42 s	**23,436**	**253**	**39,051**	520, 520, 520, 520, 520	**4 m 43 s**	0%	3,600*	**2 m 50 s**	80.10%/ 57.71%
L3	34,848	793	53,417	520, 520, 520, 520, 520	2 h	2.94%	4,080[1]	5 m 13 s	**19,728**	**217**	**35,201**	520, 520, 520, 520, 520	2 h	2.94%	4,080[1]	**1 m 5 s**	0%/ 79.23%
L4	33,912	649	48,018	520, 360, 240, 200, 400, 770	48 m 14 s	0%	3,360*	16 m 7 s	**23,112**	**217**	**34,338**	520, 360, 240, 200, 400, 770	**6 m 23 s**	0%	3,360*	**22.55 s**	86.77%/ 97.67%
L5	42,588	757	62,068	520, 360, 240, 200, 400, 770	2 h	4%	3,000[1]	8 m 19 s	**27,468**	**253**	**43,084**	520, 360, 240, 200, 400, 770	2 h	4%	3,000[1]	**1 m 54 s**	0%/ 77.15%
L6	38,304	793	56,874	520, 360, 240, 200, 400, 770	2 h	3.57%	3,360[1]	6 m 21 s	**23,184**	**217**	**38,688**	520, 360, 240, 200, 400, 770	**1 h 47 m 55 s**	**0%**	**3,360***	**24.77 s**	10.07%/ 93.50%
L7	37,368	649	51,475	320, 360, 320, 440, 480, 490, 200	2 h	1.61%	3,720[1]	16 m 32 s	**26,568**	**217**	**37,795**	320, 360, 320, 440, 480, 490, 200	**1 h 29 m 2s**	**0%**	**3,720***	**1 m 15 s**	25.81%/ 92.44%
L8	46,620	757	66,101	320, 360, 320, 440, 480, 490, 200	2 h	3.57%	3,360[1]	10 m 36 s	**31,500**	**253**	**47,117**	320, 360, 320, 440, 480, 490, 200	2 h	3.57%	3,360[1]	**2 m 24 s**	0%/ 77.36%
L9	41,760	793	60,331	320, 360, 320, 440, 480, 490, 200	2 h	3.23%	3,720[1]	6 m 9 s	**26,640**	**217**	**42,115**	320, 360, 320, 440, 480, 490, 200	**1 h 18 m 41 s**	**0%**	**3,720***	**1 m 31 s**	34.43%/ 75.34%
																Average	31.60%/ 82.06%

*Optimal solution. [1]The incumbent solution from the model. The better values are in bold.

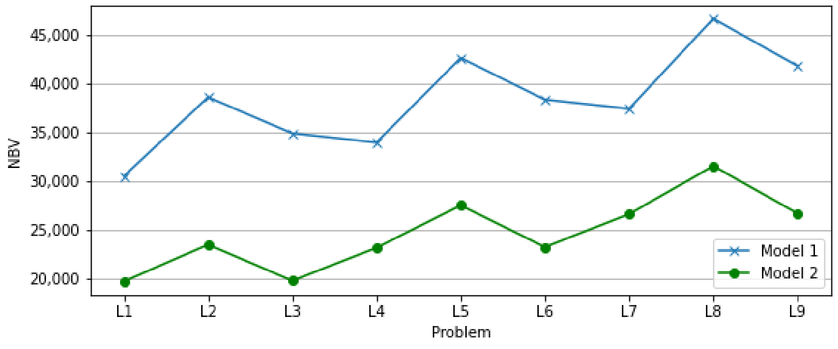


Figure 4.12: NBV comparison of Models 1 and 2 for the large problems.

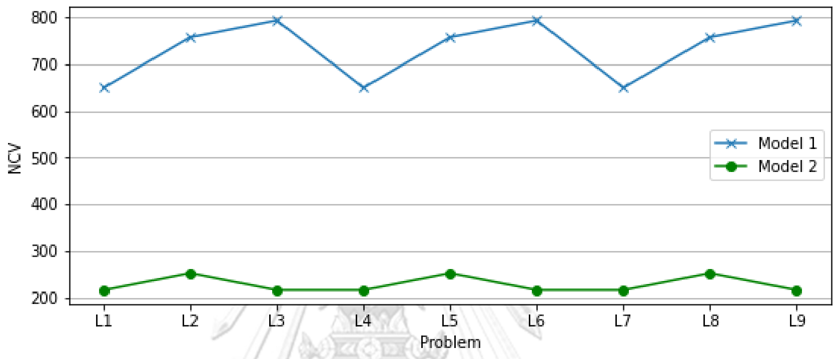


Figure 4.13: NCV comparison of Models 1 and 2 for the large problems.

%gap values from both models were also the same in each problem. Nevertheless, to reach the incumbent solution, Model 2 used smaller time$_{\text{inc}}$ than Model 1 for these problems.

The CPU time and time$_{\text{inc}}$ for solving each large problem using Models 1 and 2 are shown in Figures 4.15 and 4.16, respectively. As shown in Figure 4.15, the CPU times for solving Problems L3, L5, and L8 using Models 1 and 2 overlapped at 2 hours because both models could not solve these problems to reach optimality, but a significant difference in the CPU times between Models 1 and 2 was noticeable for the other problems. Furthermore, from Figure 4.16, the time$_{\text{inc}}$ to reach the incumbent solution for each large problem using Model 2 was lower than that using Model 1. Model 2 yielded better CPU time with average RPI$_{\text{time}}$ of 31.60% and better time$_{\text{inc}}$ with average $\text{RPI}_{\text{time}}^{\text{inc}}$ of 82.06% for the large problems. These show that Model 2 is also better than Model 1 in terms of the computational complexity for the large problems.

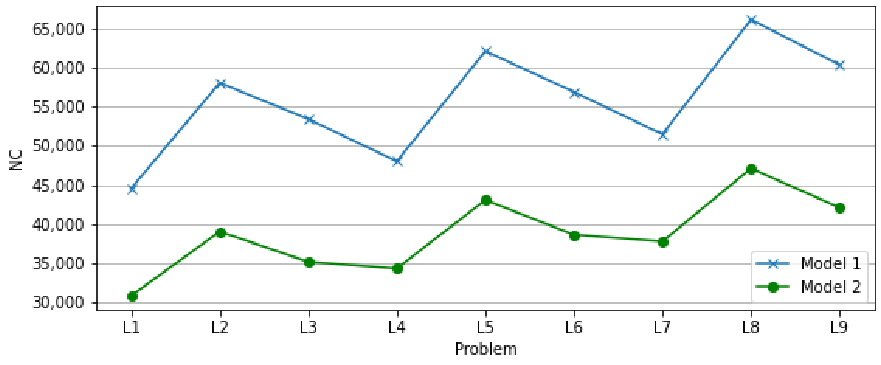


Figure 4.14: NC comparison of Models 1 and 2 for the large problems.

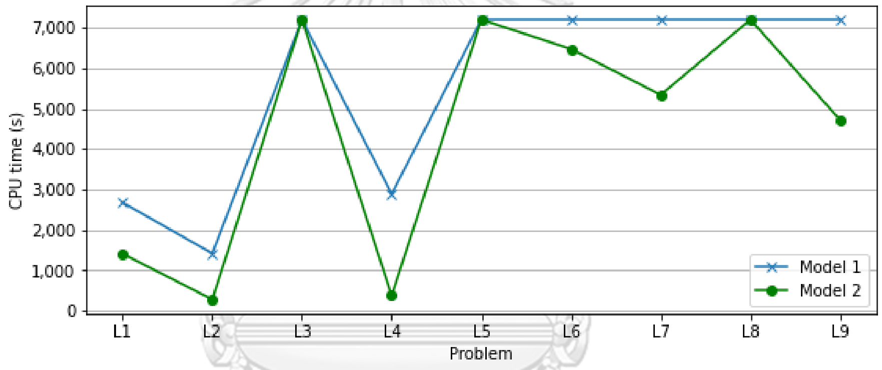


Figure 4.15: CPU time comparison of Models 1 and 2 for the large problems.

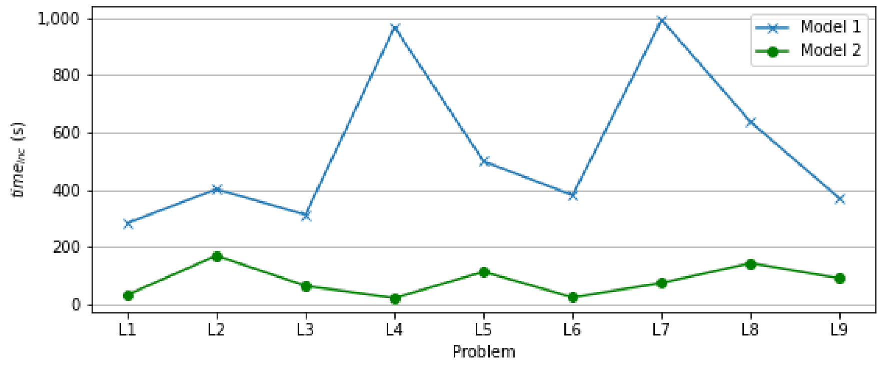


Figure 4.16: Time$_{\text{inc}}$ comparison of Models 1 and 2 for the large problems.

4.3.4 Computational results of Model 2 for the extra-large problems

The model size and computational results of Models 1 and 2 for the extra-large problems are shown in Table 4.18. Model 2 has fewer NBV, NCV, and NC as opposed to Model 1 in all extra-large problems. Figures 4.17 – 4.19 show NBV, NCV, and NC, respectively, that each model produced in each extra-large problem. It shows that, for the extra-large problems, Model 2 is better than Model 1 in terms of the size complexity.

Moreover, from Table 4.18, Model 2 could solve Problem E1 to reach optimality within the time limit, while Model 1 could not. Both models could not solve Problems E2 – E9 to reach optimality within the time limit. For Problem E2, the incumbent makespans from both models were the same, but the %gap from Model 2 (4.39%) was less than that from Model 1 (5.26%). For Problem E8, Model 2 could find an incumbent solution ($C_{\max}$ = 4,440 min), which is better than that ($C_{\max}$ = 4,560 min) by Model 1. For Problems E3 – E7 and E9, the incumbent makespans from both models were the same. However, to reach the incumbent solution, Model 2 used less time$_{\text{inc}}$ than Model 1 for these problems.

The CPU time required to solve each extra-large problem using each model was shown in Figure 4.20. The CPU time required to solve Problem E1 using Model 2 was less than that using Model 1, while the CPU times required to solve the other problems using both models overlapped at 2 hours since they could not be solved to reach optimality within the time limit. Nevertheless, Figure 4.21 shows that Model 2 used smaller time$_{\text{inc}}$ to reach the incumbent solution for all extra-large problems. Compared with Model 1, hence, Model 2 could save a lot of CPU time to reach the incumbent solution, where the incumbent solution from Model 2 was of no worse quality than that from Model 1. In particular, Model 2 used a time$_{\text{inc}}$ of 14 minutes 16 seconds to reach the incumbent solution ($C_{\max}$ = 4,440 minutes) in Problem E8, compared to 44 minutes 4 seconds ($C_{\max}$ = 4,560 minutes) for Model 1. For the other problems, the time$_{\text{inc}}$ used to reach the incumbent solution using Model 2 is only 1 – 4 minutes. The difference in time$_{\text{inc}}$ between Model 1 and Model 2 is especially notable in Problem E2 and E5. In Problem E2, Model 2 used

Table 4.18: Computational results of Models 1 and 2 for the extra-large problems.

No.	Results of Model 1								Results of Model 2								$\mathbf{RPI_{time}}$/ $\mathbf{RPI_{time}^{inc}}$
	Model Size			Results					Model Size			Results					
	NBV	NCV	NC	Outputs	CPU Time	%gap	$C_{\max}$ (min)	$\mathrm{Time_{inc}}$	NBV	NCV	NC	Outputs	CPU Time	%gap	$C_{\max}$ (min)	$\mathrm{Time_{inc}}$	
E1	46,368	865	70,937	680, 560, 760, 880, 480	2 h	4.55%	5,280[1]	12 m 49 s	**27,168**	**289**	**46,937**	680, 560, 760, 880, 480	**1 h 2 m 58 s**	**0%**	**5,160***	**1 m 13 s**	47.53%/ 90.51%
E2	59,472	1,009	93,511	680, 560, 760, 880, 480	2 h	5.26%	4,560[1]	48 m 35 s	**32,592**	**337**	**60,135**	680, 560, 760, 880, 480	2 h	**4.39%**	4,560[1]	**3 m 43 s**	0%/ 92.35%
E3	54,144	1,057	86,585	680, 560, 760, 880, 480	2 h	2.33%	5,160[1]	16 m 4 s	**27,264**	**289**	**54,617**	680, 560, 760, 880, 480	2 h	2.33%	5,160[1]	**1 m 19 s**	0%/ 91.80%
E4	50,976	865	75,546	400, 520, 800, 640, 720, 840	2 h	2.17%	5,520[1]	10 m 30 s	**31,776**	**289**	**51,546**	400, 520, 800, 640, 720, 840	2 h	2.17%	5,520[1]	**1 m 44 s**	0%/ 83.49%
E5	64,848	1,009	98,888	400, 520, 800, 640, 720, 840	2 h	2.50%	4,800[1]	1 h 18 m 26 s	**37,968**	**337**	**65,512**	400, 520, 800, 640, 720, 840	2 h	2.50%	4,800[1]	**2 m 32 s**	0%/ 96.77%
E6	58,752	1,057	91,194	400, 520, 800, 640, 720, 840	2 h	2.17%	5,520[1]	14 m 59 s	**31,872**	**289**	**59,226**	400, 520, 800, 640, 720, 840	2 h	2.17%	5,520[1]	**1 m 11 s**	0%/ 92.10%
E7	55,584	865	80,115	520, 440, 600, 400, 360, 700, 600	2 h	2.33%	5,160[1]	16 m 9 s	**36,384**	**289**	**56,155**	520, 440, 600, 400, 360, 700, 600	2 h	2.33%	5,160[1]	**1 m 18 s**	0%/ 91.95%
E8	70,224	1,009	104,265	520, 440, 600, 400, 360, 700, 600	2 h	5.26%	4,560[1]	44 m 4 s	**43,344**	**337**	**70,889**	520, 440, 600, 400, 360, 700, 600	2 h	**2.70%**	**4,440[1]**	**14 m 16 s**	0%/ 67.62%
E9	63,360	1,057	95,803	520, 440, 600, 400, 360, 700, 600	2 h	2.33%	5,160[1]	13 m 30 s	**36,480**	**289**	**63,835**	520, 440, 600, 400, 360, 700, 600	2 h	2.33%	5,160[1]	**1 m 7 s**	0%/ 91.72%
																Average	5.28%/ 88.70%

*Optimal solution. [1]The incumbent solution from the model. The better values are in bold.

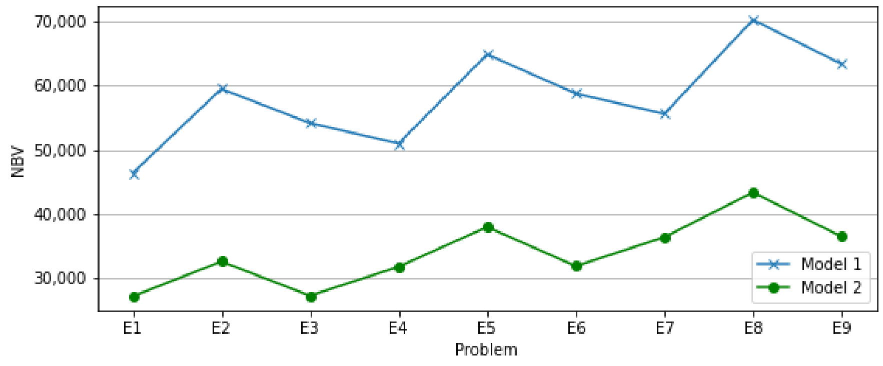


Figure 4.17: NBV comparison of Models 1 and 2 for the extra-large problems.

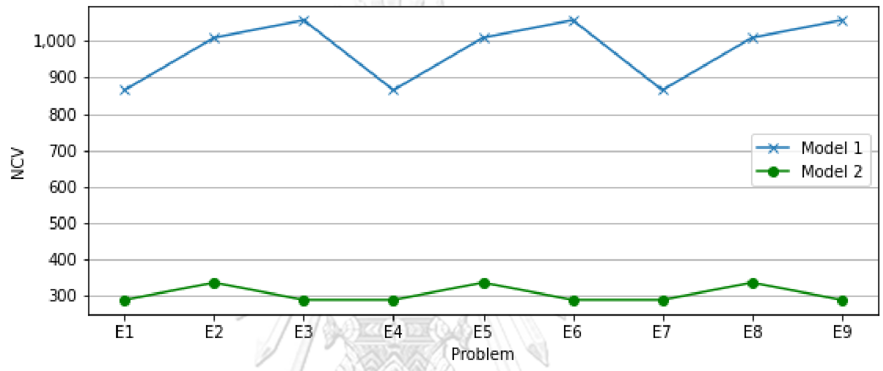


Figure 4.18: NCV comparison of Models 1 and 2 for the extra-large problems.

a time$_{\text{inc}}$ of only 3 minutes 43 seconds to reach the incumbent solution, while Model 1 used a time$_{\text{inc}}$ of 48 minutes 35 seconds to reach the incumbent solution with the same makespan. Model 2 used a time$_{\text{inc}}$ of only 2 minutes 32 seconds to reach the incumbent solution of Problem E5 compared to 78 minutes 26 seconds for Model 1 with the same makespan. Moreover, Model 2 had superior CPU time with average RPI$_{\text{time}}$ of 5.28% and superior time$_{\text{inc}}$ with average $\text{RPI}^{\text{inc}}_{\text{time}}$ of 88.70%. These show that, for the extra-large problems, Model 2 is also better than Model 1 in terms of the computational complexity.

From all numerical experiments of both models, Model 2 was found to outperform Model 1 in terms of all three factors of the size complexity because NBV, NCV, and NC that Model 2 used for each problem were less than that Model 1 used, but Model 2 is still equivalent to Model 1. For the problems that both models could solve optimally, Model 2 used a smaller computational time than Model 1 to find an optimal solution.

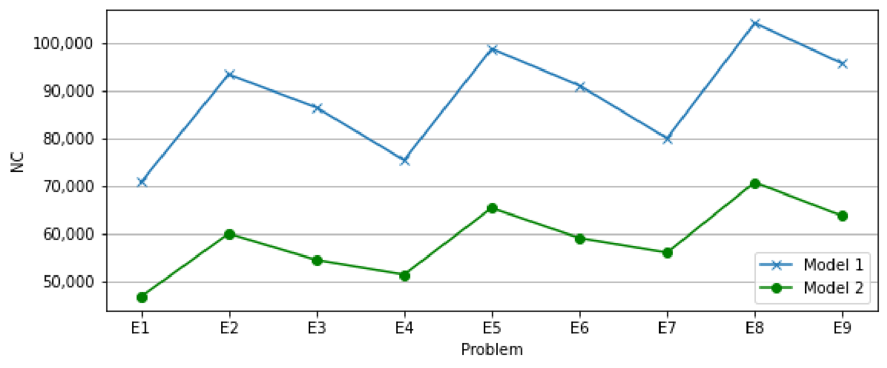


Figure 4.19: NC comparison of Models 1 and 2 for the extra-large problems.

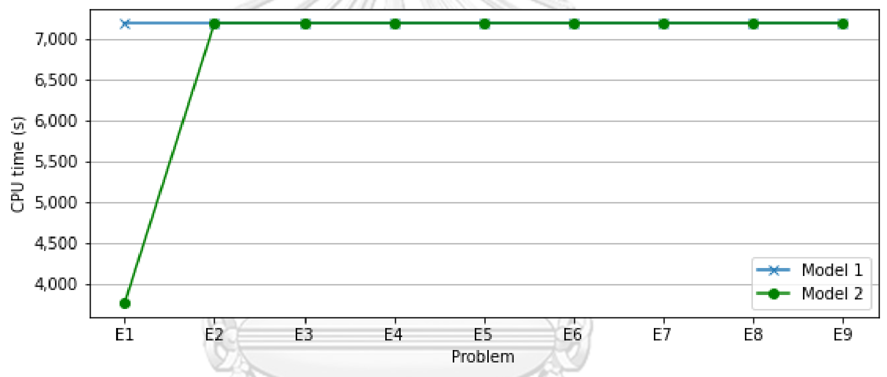


Figure 4.20: CPU time comparison of Models 1 and 2 for the extra-large problems.

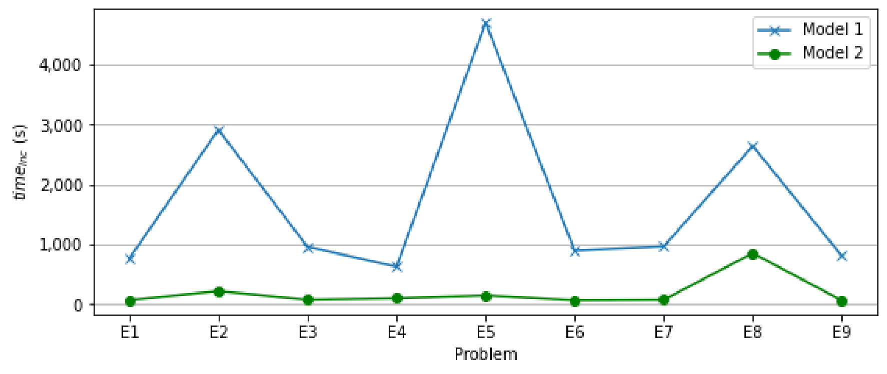


Figure 4.21: Time$_{\text{inc}}$ comparison of Models 1 and 2 for the extra-large problems.

An optimal solution for some large and extra-large problems could also be newly verified by Model 2. The average RPI_{time} of all problems (small to extra-large problems) is $\frac{5(36.84)+8(69.98)+9(31.60)+9(5.28)}{5+8+9+9} = 34.71\%$. For the problems that both models could not solve optimally within the 2-h time limit, Model 2 used a smaller computational time to find the incumbent solution, where the incumbent makespan from Model 2 is superior to or equal to the incumbent makespan from Model 1. The average $\text{RPI}^{\text{inc}}_{\text{time}}$ of the large and extra-large problems were 82.06% and 88.70%, respectively, which are very high. Therefore, it could be concluded that Model 2 also outperformed Model 1 in terms of the computational complexity. Note that the time_{inc} of each large problem and extra-large problem using Model 2 was acceptable in real-life applications. The time_{inc} of all large problems and extra-large problems using Model 2 are less than 3 minutes and 15 minutes, respectively. This shows that Model 2 is capable of finding a good solution to large-sized problems in a short amount of time. In practice, if a problem takes a long computational time to solve, a PCB manufacturer may not need to find an optimal solution. Instead, the manufacturer would prefer to find a good solution for the problem as quickly as possible. Thus, Model 2 can satisfy this preference and is a practicable option for providing a well quality schedule for the pressing process in any PCB manufacturing industry.

4.4 Computational results of the 3P-PCB-PH algorithm

In this section, the 3P-PCB-PH algorithm implemented in Python 3.7.3 was used to solve all test problems on the same hardware environment as in the previous section. To capture the variation in the computational time, each problem was run 10 times. The results from the 3P-PCB-PH algorithm were compared with the results from Models 1 and 2.

4.4.1 Computational results of the 3P-PCB-PH algorithm for the small problems

Table 4.19 shows the results of the 3P-PCB-PH algorithm as well as Models 1 and 2 for the small problems. The results from the 3P-PCB-PH algorithm consisted of the number of finished goods of each panel type (outputs), the makespan, and the average

Table 4.19: Computational results of the 3P-PCB-PH algorithm and Models 1 and 2 for the small problems.

No.	**Results of Model 1**			**Results of Model 2**			**Results of 3P-PCB-PH Algorithm**		
	Outputs	$C_{\max}$	**CPU Time**	**Outputs**	$C_{\max}$	**CPU Time**	**Outputs**	$C_{\max}$	**Avg CPU Time (SD)**
S1	120, 120, 160	1,440*	2.31 s	120, 120, 160	1,440*	1.51 s	120, 120, 160	1,440*	**0.00349 s (0.00085 s)**
S2	200, 240, 240	2,160*	2.48 s	200, 240, 240	2,160*	1.76 s	200, 240, 240	2,160*	**0.00488 s (0.00246 s)**
S3	280, 280, 240	2,520*	3.21 s	280, 280, 240	2,520*	1.82 s	280, 280, 240	2,520*	**0.00658 s (0.00346 s)**
S4	120, 160, 160	1,200*	8.15 s	120, 160, 160	1,200*	5.29 s	120, 160, 160	1,200*	**0.00598 s (0.00342 s)**
S5	120, 160, 160	1,080*	3.18 s	120, 160, 160	1,080*	1.84 s	120, 160, 160	1,080*	**0.00509 s (0.00371 s)**

*Optimal solution. The better values are in bold.

CPU time over 10 runs (Avg CPU time). The results show that each small problem could be solved to a feasible solution by the 3P-PCB-PH algorithm, where the makespan of the solution from the 3P-PCB-PH algorithm is equal to the optimal makespan from both Models 1 and 2, i.e., the 3P-PCB-PH algorithm could also find an optimal solution for all small problems. The number of outputs of each panel type satisfied the demand in each problem. Although the small problems could be solved by Models 1 and 2 using only a small CPU time (2 – 9 seconds for Model 1 and 1 – 6 seconds for Model 2), these problems could be solved by the 3P-PCB-PH algorithm using less CPU time than both Models 1 and 2. On average, the 3P-PCB-PH algorithm could solve each small problem in less than 0.01 seconds, and the standard deviation (SD) of CPU time is less than 0.01 seconds. This shows that the 3P-PCB-PH algorithm is effective and efficient.

4.4.2 Computational results of the 3P-PCB-PH algorithm for the medium problems

The results of the 3P-PCB-PH algorithm as well as Models 1 and 2 for the medium problems are compared in Table 4.20. The results show that the 3P-PCB-PH algorithm could find a feasible solution with the same makespan as the optimal solution from Model 1 and Model 2 for each problem, i.e., the 3P-PCB-PH algorithm could also solve all medium problems to an optimal solution. From Table 4.20, although the CPU times for solving the medium problems using Model 2 were lower than that using Model 1, the 3P-PCB-PH algorithm still used lower CPU time than Model 2 for solving each medium problem. On average, the 3P-PCB-PH algorithm could solve each medium problem in less than 0.01 seconds and with SD of CPU time of less than 0.01 seconds. This shows the effectiveness and efficiency of the 3P-PCB-PH algorithm.

4.4.3 Computational results of the 3P-PCB-PH algorithm for the large problems

Table 4.21 shows the results of the 3P-PCB-PH algorithm as well as Models 1 and 2 for the large problems. For Problems L1, L2, and L4, the 3P-PCB-PH algorithm could find an optimal solution because the makespan of the solution from the 3P-PCB-PH

Table 4.20: Computational results of the 3P-PCB-PH algorithm and Models 1 and 2 for the medium problems.

No.	**Results of Model 1**			**Results of Model 2**			**Results of 3P-PCB-PH Algorithm**		
	Outputs	$C_{\max}$	**CPU Time**	**Outputs**	$C_{\max}$	**CPU Time**	**Outputs**	$C_{\max}$	**Avg CPU Time (SD)**
M1	320, 320, 320	1,560*	32.81 s	320, 320, 320	1,560*	13.93 s	320, 320, 320	1,560*	**0.00469 s (0.00141 s)**
M2	480, 480, 520	2,520*	16.79 s	480, 480, 520	2,520*	6.29 s	480, 480, 520	2,520*	**0.00519 s (0.00248 s)**
M3	720, 920, 600	3,600*	1 m 28 s	720, 920, 600	3,600*	25.46 s	720, 920, 600	3,600*	**0.00658 s (0.00245 s)**
M4	200, 320, 400, 120	1,800*	20.65 s	200, 320, 400, 120	1,800*	2.40 s	200, 320, 400, 120	1,800*	**0.00599 s (0.00266 s)**
M5	320, 400, 200, 520	2,280*	3 m 59 s	320, 400, 200, 520	2,280*	55.21 s	320, 400, 200, 520	2,280*	**0.00768 s (0.00342 s)**
M6	520, 720, 720, 520	3,960*	4 m 2 s	520, 720, 720, 520	3,960*	20.48 s	520, 720, 720, 520	3,960*	**0.00927 s (0.00509 s)**
M7	200, 280, 200, 280, 200	1,920*	9 m 31 s	200, 280, 200, 280, 200	1,920*	6 m 59 s	200, 280, 200, 280, 200	1,920*	**0.00768 s (0.00282 s)**
M8	400, 320, 200, 280, 320	2,520*	49.31 s	400, 320, 200, 280, 320	2,520*	7.26 s	400, 320, 200, 280, 320	2,520*	**0.00909 s (0.00331 s)**

*Optimal solution. The better values are in bold.

Table 4.21: Computational results of the 3P-PCB-PH algorithm and Models 1 and 2 for the large problems.

No.	Results of Model 1				Results of Model 2				Results of 3P-PCB-PH Algorithm		
	Outputs	**$C_{\max}$ (%gap)**	**CPU Time**	**Time$_{\text{inc}}$**	**Outputs**	**$C_{\max}$ (%gap)**	**CPU Time**	**Time$_{\text{inc}}$**	**Outputs**	**$C_{\max}$**	**Avg CPU Time (SD)**
L1	520, 520, 520, 520, 520	4,080* (0%)	44 m 38 s	4 m 44 s	520, 520, 520, 520, 520	4,080* (0%)	23 m 34 s	33.68 s	520, 520, 520, 520, 520	4,080*	**0.00928 s (0.00346 s)**
L2	520, 520, 520, 520, 520	3,600* (0%)	23 m 42 s	6 m 42 s	520, 520, 520, 520, 520	3,600* (0%)	4 m 43 s	2 m 50 s	520, 520, 520, 520, 520	3,600*	**0.00918 s (0.00297 s)**
L3	520, 520, 520, 520, 520	4,080[1] (2.94%)	2 h	5 m 13 s	520, 520, 520, 520, 520	4,080[1] (2.94%)	2 h	1 m 5 s	520, 520, 520, 520, 520	4,080	**0.00987 s (0.00447 s)**
L4	520, 360, 240, 200, 400, 770	3,360* (0%)	48 m 14 s	16 m 7s	520, 360, 240, 200, 400, 770	3,360* (0%)	6 m 23 s	22.55 s	520, 360, 240, 200, 400, 770	3,360*	**0.00997 s (0.00326 s)**
L5	520, 360, 240, 200, 400, 770	3,000[1] (4%)	2h	8 m 19 s	520, 360, 240, 200, 400, 770	3,000[1] (4%)	2 h	1 m 54 s	520, 360, 240, 200, 400, 770	3,000	**0.00908 s (0.00291 s)**
L6	520, 360, 240, 200, 400, 770	3,360[1] (3.57%)	2h	6 m 21 s	520, 360, 240, 200, 400, 770	3,360* (0%)	1 h 47 m 55 s	24.77 s	520, 360, 240, 200, 400, 770	3,360*	**0.00993 s (0.00575 s)**
L7	320, 360, 320, 440, 480, 490, 200	3,720[1] (1.61%)	2h	16 m 32 s	320, 360, 320, 440, 480, 490, 200	3,720* (0%)	1 h 29 m 2 s	1 m 15 s	320, 360, 320, 440, 480, 490, 200	3,720*	**0.01015 s (0.00319 s)**
L8	320, 360, 320, 440, 480, 490, 200	3,360[1] (3.57%)	2h	10 m 36 s	320, 360, 320, 440, 480, 490, 200	3,360[1] (3.57%)	2 h	2 m 24 s	320, 360, 320, 440, 480, 490, 200	3,360	**0.01250 s (0.00504 s)**
L9	320, 360, 320, 440, 480, 490, 200	3,720[1] (3.23%)	2h	6 m 9 s	320, 360, 320, 440, 480, 490, 200	3,720* (0%)	1 h 18 m 41 s	1 m 31 s	320, 360, 320, 440, 480, 490, 200	3,720*	**0.01057 s (0.00566 s)**

*Optimal solution. [1]The incumbent solution from the MILP model. The better values are in bold.

algorithm is equal to the optimal makespan from both MILP models. Note that the CPU times for solving Problems L1, L2, and L4 optimally using Model 1 are large, and the CPU times for solving these problems using Model 2 are less than that using Model 1. Nevertheless, the 3P-PCB-PH algorithm could find an optimal solution for these problems using an average and SD of CPU time of less than 0.01 seconds each. For Problems L6, L7, and L9, the optimality could be verified by Model 2, which took a large CPU time (greater than 1 hour) in each problem. Although the time$_{\text{inc}}$ for solving these problems using Model 2 is small (the maximum value is 1 minute 31 seconds in Problem L9) and less than that using Model 1, the 3P-PCB-PH algorithm could still find an optimal solution for these problems using an average and SD of CPU time of less than 0.1 seconds each. For Problems L3, L5, and L8, Models 1 and 2 could not verify optimality but could find an incumbent solution for each problem with the same makespan within the 2-h time limit. The time$_{\text{inc}}$ to reach the incumbent solution for each problem using Model 2 is small and less than that using Model 1, but the 3P-PCB-PH algorithm could find a solution with the same makespan as the incumbent solution from each model for each problem using a very small CPU time. Both the average and SD of CPU time for solving Problems L3, L5, and L8 using the 3P-PCB-PH algorithm are only less than 0.1 seconds. These results demonstrate that the 3P-PCB-PH algorithm is very effective and efficient.

The following is an example of a solution from the 3P-PCB-PH algorithm. For the results of Problem L1, variables x_{iklpt} and X_{pto}, which were equal to 1, are shown in Tables 4.22 and 4.23, respectively. Figures 4.22 and 4.23 show the Gantt charts of the press machine and oven, respectively. These Gantt charts were different from the Gantt charts in Figures 4.2 and 4.3 from Model 1, i.e., Problem L1 has an alternative optimal schedule. The makespan of the overall process of this problem was 4,080 minutes. It is worth noting that, in Figure 4.22, each panel type is finished in a group, which is preferred in the real world because it makes material preparation easier for the PCB manufacturer.

Table 4.22: List of x_{iklpt} values which is equal to 1 in the optimal solution of Problem L1 using the 3P-PCB-PH algorithm.

Press Machine	**Non-zero x_{iklpt}**
1	$x_{11211}, x_{11212}, x_{11213}, x_{23214}, x_{23215}, x_{32216}, x_{32217}, x_{43218}, x_{43219}, x_{5121,10}, x_{5121,11}$
2	$x_{11221}, x_{11222}, x_{23223}, x_{23224}, x_{23225}, x_{32226}, x_{32227}, x_{43228}, x_{43229}, x_{5122,10}, x_{5122,11}$
3	$x_{11231}, x_{11232}, x_{23233}, x_{23234}, x_{32235}, x_{32236}, x_{32237}, x_{43238}, x_{43239}, x_{5123,10}, x_{5123,11}$
4	$x_{11241}, x_{11242}, x_{23243}, x_{23244}, x_{32245}, x_{32246}, x_{43247}, x_{43248}, x_{43249}, x_{5124,10}, x_{5124,11}$
5	$x_{11251}, x_{11252}, x_{23253}, x_{23254}, x_{32255}, x_{32256}, x_{43257}, x_{43258}, x_{51259}, x_{5125,10}, x_{5125,11}$
6	$x_{11261}, x_{11262}, x_{23263}, x_{23264}, x_{32265}, x_{32266}, x_{43267}, x_{43268}, x_{51269}, x_{5126,10}$

Table 4.23: List of X_{pto} values which is equal to 1 in the optimal solution of Problem L1 using the 3P-PCB-PH algorithm.

Press Machine	**Non-zero X_{pto}**
1	$X_{111}, X_{121}, X_{131}, X_{141}, X_{151}, X_{161}, X_{171}, X_{181}, X_{191}, X_{1,10,1}, X_{1,11,1}$
2	$X_{212}, X_{222}, X_{232}, X_{242}, X_{252}, X_{262}, X_{272}, X_{282}, X_{292}, X_{2,10,2}, X_{2,11,2}$
3	$X_{313}, X_{323}, X_{333}, X_{343}, X_{353}, X_{363}, X_{373}, X_{383}, X_{393}, X_{3,10,3}, X_{3,11,3}$
4	$X_{411}, X_{421}, X_{431}, X_{441}, X_{451}, X_{461}, X_{471}, X_{481}, X_{491}, X_{4,10,1}, X_{4,11,1}$
5	$X_{512}, X_{522}, X_{532}, X_{542}, X_{552}, X_{562}, X_{572}, X_{582}, X_{592}, X_{5,10,2}, X_{5,11,2}$
6	$X_{613}, X_{623}, X_{633}, X_{643}, X_{653}, X_{663}, X_{673}, X_{683}, X_{693}, X_{6,10,3}$

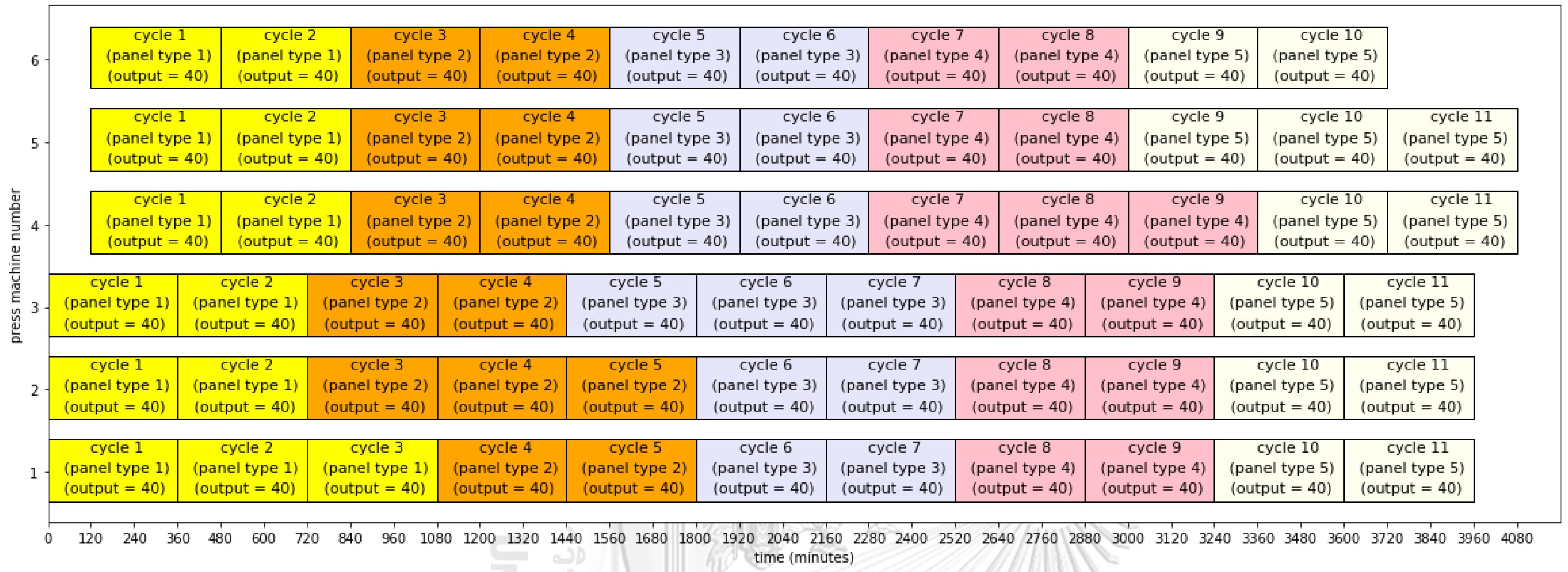


Figure 4.22: Gantt chart of press machines for Problem L1 using the 3P-PCB-PH algorithm.

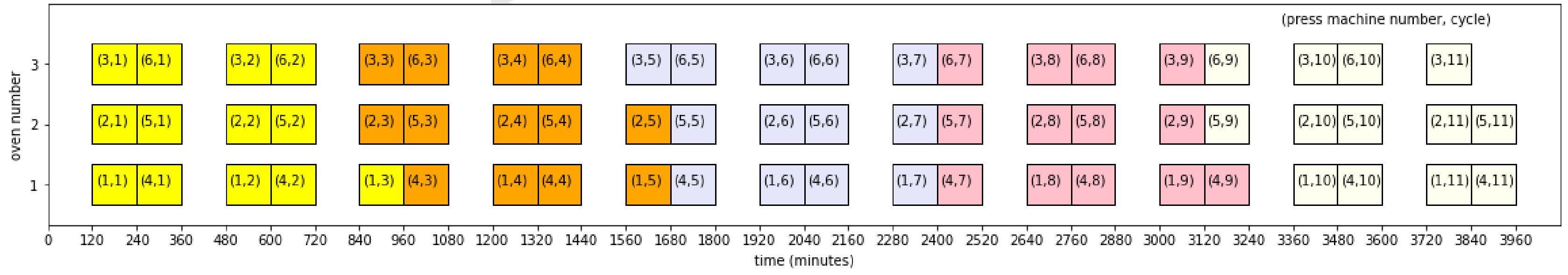


Figure 4.23: Gantt chart of ovens for Problem L1 using the 3P-PCB-PH algorithm.

4.4.4 Computational results of the 3P-PCB-PH algorithm for the extra-large problems

Table 4.24 shows the results of extra-large problems from the 3P-PCB-PH algorithm compared with the results from Models 1 and 2. For Problem E1, the 3P-PCB-PH algorithm could find an optimal solution since the makespan of the solution from the 3P-PCB-PH algorithm is equal to the optimal makespan from Model 2. The solution from the 3P-PCB-PH algorithm ($C_{\max} = 5,160$) is also better than the solution from Model 1 ($C_{\max} = 5,280$). However, the CPU time for solving Problem E1 optimally using Model 2 is large (1 hour 2 minutes 58 seconds). The 3P-PCB-PH algorithm could find an optimal solution for this problem using a small average and SD of CPU time of less than 0.1 seconds each, and this Avg CPU time is also less than the time$_{\text{inc}}$ value using Model 2. For Problem E8, the 3P-PCB-PH algorithm could find a feasible solution with the same makespan as the incumbent solution from Model 2 ($C_{\max} = 4,440$), which is better than the incumbent solution from Model 1 ($C_{\max} = 4,560$). The 3P-PCB-PH algorithm used Avg CPU time (and SD CPU time) of less than 0.1 seconds for solving this problem, which is much less than the time$_{\text{inc}}$ value using Model 2. For Problems E2 – E7, and E9, the 3P-PCB-PH algorithm could find a solution with the same makespan as the incumbent solution from both Models 1 and 2, but the 3P-PCB-PH algorithm used less CPU time than the time$_{\text{inc}}$ value from both Models 1 and 2. Both the average and SD of CPU time for solving Problems E2 – E7 and E9 using the 3P-PCB-PH algorithm are only less than 0.1 seconds. These results show the effectiveness and efficiency of the proposed 3P-PCB-PH algorithm.

Furthermore, the results from the 3P-PCB-PH algorithm can give helpful information. For example, from Problems E1 – E3 in Table 4.8, all of the parameters in these problems are the same, with the exception of the number of press machines and ovens, which were increased by one in Problems E2 and E3, respectively, from Problem E1. According to the results of these problems in Table 4.24, the makespan of Problems E1 – E3 were 5,160 minutes, 4,560 minutes, and 5,160 minutes, respectively. This means that the makespan of Problem E1 could be reduced from 5,160 to 4,560 minutes when the num-

Table 4.24: Computational results of the 3P-PCB-PH algorithm and Models 1 and 2 for the extra-large problems.

No.	**Results of Model 1**				**Results of Model 2**				**Results of 3P-PCB-PH Algorithm**		
	Outputs	$C_{\max}$ **(%gap)**	**CPU Time**	**Time$_{\text{inc}}$**	**Outputs**	$C_{\max}$ **(%gap)**	**CPU Time**	**Time$_{\text{inc}}$**	**Outputs**	$C_{\max}$	**Avg CPU Time (SD)**
E1	680, 560, 760, 880, 480	5,280^1 (4.55%)	2 h	12 m 49 s	680, 560, 760, 880, 480	5,160* (0%)	1 h 2 m 58 s	1 m 13 s	680, 560, 760, 880, 480	5,160*	**0.01315 s (0.00508 s)**
E2	680, 560, 760, 880, 480	4,560^1 (5.26%)	2 h	48 m 35 s	680, 560, 760, 880, 480	4,560^1 (4.39%)	2 h	3 m 43 s	680, 560, 760, 880, 480	4,560	**0.01371 s (0.00369 s)**
E3	680, 560, 760, 880, 480	5,160^1 (2.33%)	2 h	16 m 4 s	680, 560, 760, 880, 480	5,160^1 (2.33%)	2 h	1 m 19 s	680, 560, 760, 880, 480	5,160	**0.01440 s (0.00545 s)**
E4	400, 520, 800, 640, 720, 840	5,520^1 (2.17%)	2 h	10 m 30 s	400, 520, 800, 640, 720, 840	5,520^1 (2.17%)	2 h	1 m 44 s	400, 520, 800, 640, 720, 840	5,520	**0.01526 s (0.00491 s)**
E5	400, 520, 800, 640, 720, 840	4,800^1 (2.50%)	2h	1 h 18 m 26 s	400, 520, 800, 640, 720, 840	4,800^1 (2.50%)	2 h	2 m 32 s	400, 520, 800, 640, 720, 840	4,800	**0.01684 s (0.00845 s)**
E6	400, 520, 800, 640, 720, 840	5,520^1 (2.17%)	2h	14 m 59 s	400, 520, 800, 640, 720, 840	5,520^1 (2.17%)	2 h	1 m 11 s	400, 520, 800, 640, 720, 840	5,520	**0.01559 s (0.00690 s)**
E7	520, 440, 600, 400, 360, 700, 600	5,160^1 (2.33%)	2h	16 m 9 s	520, 440, 600, 400, 360, 700, 600	5,160^1 (2.33%)	2 h	1 m 18 s	520, 440, 600, 400, 360, 700, 600	5,160	**0.01677 s (0.00535 s)**
E8	520, 440, 600, 400, 360, 700, 600	4,560^1 (5.26%)	2h	44 m 4 s	520, 440, 600, 400, 360, 700, 600	4,440^1 (2.70%)	2 h	14 m 16 s	520, 440, 600, 400, 360, 700, 600	4,440	**0.01714 s (0.00560 s)**
E9	520, 440, 600, 400, 360, 700, 600	5,160^1 (2.33%)	2h	13 m 30 s	520, 440, 600, 400, 360, 700, 600	5,160^1 (2.33%)	2 h	1 m 7 s	520, 440, 600, 400, 360, 700, 600	5,160	**0.01735 s (0.00648 s)**

*Optimal solution. ^{1}The incumbent solution from the MILP model. The better values are in bold.

ber of press machines is increased by one, but the makespan could not be reduced when the number of ovens is increased by one from Problem E1. Therefore, the 3P-PCB-PH algorithm can also be used to determine which resources should be increased in order to reduce the production time.

From all numerical experiments, the results show that the 3P-PCB-PH algorithm is appropriate for solving all sizes of problems. For the problems that Model 1 or Model 2 could solve optimally, it could find an optimal solution for these problems using a much less computational time. For the other problems that both models could not solve optimally, the 3P-PCB-PH algorithm could find a solution with the same makespan as the incumbent solution from Model 2, which is less than or equal to the makespan of the incumbent solution from Model 1. The computational times of the 3P-PCB-PH algorithm seem to be very fast due to its simplicity, where all problems could be solved within 1 second. The 3P-PCB-PH algorithm could therefore save a large amount of computational time at finding a good schedule for the pressing process scheduling and is practical for practitioners in the real PCB manufacturing industries.

4.5 Discussions

In this research, Model 1 was firstly proposed for solving the pressing process scheduling. Although this model could be used to find a solution of the pressing process scheduling, it is not effective due to its large size complexity. It used a large number of constraints and binary variables, especially in the constraints for avoiding an overlap in an oven. In addition, the model used too many continuous variables for defining the starting time of the pressing phase of a press machine cycle. Furthermore, it also used the completion time variables, which are not need to be defined in the model. These can lead to a poor model performance because the model required a large computational time to be solved. The results showed that Model 1 could solve the small and medium problems to reach optimality, while it could solve only some large problems optimally. Model 2, which is an improved version from Model 1, was then proposed. Model 2 is better than Model 1 in all three factors of the size complexity, where NBV, NCV, and NC in Model

2 are less than those in Model 1. A large number of binary variables and constraints in Model 2 were reduced from Model 1, especially in the constraints for avoiding an overlap in an oven, since Model 2 replaced the binary variable $Y_{ptp't'o}$ by $Y_{ptp't'}$ and could reduce half of NC of the constraints for avoiding an overlap in an oven. NBV of $Y_{ptp't'o}$ and $Y_{ptp't'}$ are significantly different, and NC in the constraints for avoiding an overlap in an oven from both models are quite different when the size of problem becomes large or extra-large. Since the most important factor that influences the MILP model's performance is NBV, followed by NC [17], reducing these in Model 2 can help solve large-sized problems more efficiently. The results showed that Model 2 used smaller CPU time and time_{inc} than Model 1 for solving all problems, where the solution from Model 2 was of no worse quality than that from Model 1, i.e., Model 2 also outperformed Model 1 in terms of the computational complexity.

In addition, the time_{inc} from Model 2 to reach the incumbent solution of large and extra-large problems are small and practicable. In practice, many PCB companies need to find a good schedule for the pressing process scheduling in a short amount of time, and an incumbent solution may be just what they need. Therefore, a PCB manufacturer can use Model 2 to find an optimal schedule or a good schedule for the pressing process scheduling by setting a small time limit, such as 5 minutes, or a small %gap, such as 5%.

The 3P-PCB-PH algorithm is the last approach that was presented for solving the pressing process scheduling. The main idea of this heuristic algorithm was to balance workload in all press machines and ovens. The results showed that the 3P-PCB-PH algorithm could solve problems of every size using a very small computational time because of its simplicity. The schedule from the 3P-PCB-PH algorithm is also preferable in the real situation than the schedule from both Models 1 and 2 because each panel type is finished in a single group, while cycles of the same panel type in the solution from Model 1 or Model 2 may not need to be scheduled consecutively.

The proposed MILP models have the benefit that they can guarantee to find an optimal solution if the problem could be solved to reach optimality within the time limit.

The solution from a MILP model can also be used as a reference solution to decide the quality of the solution from the proposed heuristic algorithm. On the contrary, the proposed 3P-PCB-PH algorithm has the benefit that it can save a lot of computational time to find a good solution for the problem, but it cannot guarantee to find an optimal solution.

The proposed models can be easily expanded to make them more useful in real-world applications. For example, the objective of the proposed models is to minimize the makespan of the pressing process, where the demands must be satisfied. Nevertheless, the surplus output of each panel type may be too large. The term $\epsilon \sum_{i=1}^{I} \left(\sum_{k=1}^{K} \sum_{l=1}^{L} \sum_{p=1}^{P} \sum_{t=1}^{T} x_{iklpt}(ma_{ikl}) - d_i \right)$ can be added to the objective function if we also want to enforce that the surplus output of each type of panels should not be too large with the main objective makespan. Note that the constant ϵ should be very small so that it does not affect minimizing the main objective makespan.

Furthermore, both proposed models and the proposed heuristic algorithm can be beneficial in the real-world situations as follows.

1. The proposed models or the proposed heuristic algorithm can be used to find an optimal schedule or a good schedule for the pressing process scheduling by a PCB company which manually schedules the pressing process. The solution from the proposed models or the proposed heuristic algorithm can be used to compare with the manual schedule of the PCB company, and the manager of the PCB company can select the better schedule to manage in the real situation.

2. If a PCB company receives a rush order, the proposed models or the proposed heuristic algorithm can be used to decide which resources should be increased in order to reduce the production time before executing the production plan in the real-world situation. For example, the manager of the PCB company can adjust the parameters in the proposed models or the proposed heuristic algorithm, such as the number of press machines or ovens, to see how much the makespan can be reduced.

CHAPTER V

CONCLUSIONS

This chapter summarizes the conclusion of this dissertation and provides some extensions that could be future research.

5.1 Conclusions

This dissertation studied the pressing process scheduling, a real-world application in the PCB manufacturing industry. In a cycle of a press machine, the panels are created and inserted into a press machine (lay-up phase) and then sent into an oven so that they are heated and pressed in the oven (pressing phase). After that, the pressed books are cooled down for a while in the press machine (cool-down phase). The press machines have to be processed several cycles so that the demands of panels are satisfied. The pressing process scheduling uses assignment and sequencing to obtain an optimal solution, i.e., the assignment of a panel type, a SST size, and a layout in creating a book, the assignment of a press machine to an oven, and the sequencing of tasks from press machines in an oven. The objective of the pressing process is to minimize makespan, which can imply increasing the utilization of the available resources and is the main objective of any PCB manufacturer.

In this dissertation, three approaches for solving the pressing process scheduling were proposed, which consisted of Model 1, Model 2, and the 3P-PCB-PH algorithm. Both Model 1 and Model 2 are MILP, which is an exact method, while the 3P-PCB-PH algorithm is a heuristic method. Both Model 1 and Model 2 illustrated possible applications of MILP that can cope with a complicate problem from an actual industry. The MILP model has the benefit that it can give an optimal solution of the problem if one exists, while the heuristic algorithm has the benefit that it can find a good solution of the problem within a reasonable time. In this work, we acquired real data from a PCB company, and the data was used to generate the test problems, which included the small, medium, large, and extra large problems. Model 1 could solve all small and medium problems optimally within the 2-h time limit, but it could find an optimal solution for only some large problems and could not solve all extra-large problems optimally. This is because the size complexity of Model 1 is large, where it used too many constraints and binary variables to formulate the model. This led to a poor model performance (poor computational complexity). Therefore, we aimed to develop another MILP model. The development of MILP model is challenging since the new model may verify a new optimal solution for the problems that the previous model could not solve optimally or may find a better feasible solution.

Model 2 is an improved version of Model 1, where it outperformed Model 1 in terms of both size and computational complexities. NBV, NCV, and NC in Model 2 are less than those in Model 1. It took lower computational time than Model 1 to solve all the problems that Model 1 could solve optimally and could newly verify the optimal solution for some problems that Model 1 could not. Model 2 yielded a better computational time with average $\mathrm{RPI}_{\mathrm{time}}$ of 34.71% for finding an optimal solution. Model 2 could also find an incumbent solution for the large and extra-large problems using less computational time compared with Model 1, where the incumbent solution from Model 2 is superior or of the same quality as the incumbent solution from Model 1. Model 2 also yielded a better computational time to reach the incumbent solution with average $\mathrm{RPI}_{\mathrm{time}}^{\mathrm{inc}}$ of 82.06% and 88.70% for large and extra-large problems, respectively. These show that Model 2 is suitable for finding a good solution for the large-sized pressing process scheduling problems.

As for the 3P-PCB-PH algorithm, the main idea was to balance workload in all press machines and ovens. The algorithm is simple but effective and could find an optimal solution or a near-optimal solution for the test problems using a very small computational time. The solution from the 3P-PCB-PH algorithm is of the same quality as the solution from Model 2 for all test problems, which is no worse than that from Model 1. When the size of problem increased from small size to extra-large size, the computational time of the 3P-PCB-PH algorithm are not hugely increased due to its simplicity, which is different from the computational time of the MILP models since there are a large number of feasible solutions to be verified for optimality owing to a lot of decision variables. The 3P-PCB-PH algorithm could solve all problems using the computational time of less than 1 second, which is very practical in the real situation. All proposed models and heuristic algorithm can be options to provide an optimal schedule or a high quality schedule for the pressing process in any PCB manufacturing industries to reduce their production cost and time.

The limitations of this research are that the pressing process is assume to have the same size of press machines and ovens as well as the same processing time of all panel types. Furthermore, the approach of MILP model is not suitable for solving the pressing process scheduling problems with long planning horizons, such as, a month. In addition, the proposed heuristic algorithm is specific for only the pressing process scheduling problem with the assumptions as described above and cannot be used for the problem with new additional constraints or other assumptions. However, the idea of workload balancing, assignment, and sequencing from the proposed heuristic algorithm can be applied to similar problems from other industries.

5.2 Future works

This section provides further developments that could be future research, which include the following.

1. In the pressing process, some additional constraints can be introduced, which consists of the following:
    - A press machine cycle can press multiple types of panels at the same time.
    - The panel types have different cycle times.
    - Some panel types have a higher priority to be finished before other panel types.
    - Machine maintenance is required for the press machines or ovens.

- A cycle of a press machine can leave some openings to be empty.
- There are many sizes of press machines and many sizes of ovens.

Adding these factors to the problem can be very interesting for further research, but it would also increase the complication of the problem.

2. The development of a new MILP model which can outperform the proposed models for the pressing process scheduling in this dissertation is also very challenging.
3. The development of new heuristic or metaheuristic algorithms for solving the pressing process scheduling problem can also be another future research line.

REFERENCES

[1] A. Noroozi and H. Mokhtari, "Scheduling of printed circuit board (PCB) assembly systems with heterogeneous processors using simulation-based intelligent optimization methods," *Neural. Comput. Appl.*, vol. 26, no. 4, pp. 857–873, 2015.

[2] W. Qin, Z. Zhuang, Y. Liu, and O. Tang, "A two-stage ant colony algorithm for hybrid flow shop scheduling with lot sizing and calendar constraints in printed circuit board assembly," *Comput. Ind. Eng.*, vol. 138, p. 106115, 2019.

[3] R. S. Khandpur, *Printed circuit boards: Design, fabrication, assembly and testing.* Tata McGraw-Hill Education, 2005.

[4] M. S. Bazaraa, J. J. Jarvis, and H. D. Sherali, *Linear programming and network flows.* John Wiley & Sons, 2008.

[5] D. S. Chen, R. G. Batson, and Y. Dang, *Applied integer programming: Modeling and solution.* John Wiley & Sons, 2011.

[6] I. M. Alharkan, "Algorithms for sequencing and scheduling," *Industrial Engineering Department, King Saud University, Riyadh, Saudi Arabia*, 2005.

[7] F. Pezzella, G. Morganti, and G. Ciaschetti, "A genetic algorithm for the flexible job-shop scheduling problem," *Comput. Oper. Res.*, vol. 35, no. 10, pp. 3202–3212, 2008.

[8] A. Thammano and A. Phu-Ang, "Flexible production scheduling on parallel machines in manufacturing industry," *KMITL IT J.*, vol. 2, no. 1, 2016, (in Thai).

[9] C. Özgüven, L. Özbakır, and Y. Yavuz, "Mathematical models for job-shop scheduling problems with routing and process plan flexibility," *Appl. Math. Model.*, vol. 34, no. 6, pp. 1539–1548, 2010.

[10] V. Roshanaei, A. Azab, and H. ElMaraghy, “Mathematical modelling and a meta-heuristic for flexible job shop scheduling,” *Int. J. Prod. Res.*, vol. 51, no. 20, pp. 6247–6274, 2013.

[11] G. Zhang, L. Gao, and Y. Shi, “An effective genetic algorithm for the flexible job-shop scheduling problem,” *Expert Syst. Appl.*, vol. 38, no. 4, pp. 3563–3573, 2011.

[12] X. Li and L. Gao, “An effective hybrid genetic algorithm and tabu search for flexible job shop scheduling problem,” *Int. J. Prod. Econ.*, vol. 174, pp. 93–110, 2016.

[13] B. Naderi and A. Azab, “An improved model and novel simulated annealing for distributed job shop problems,” *Int. J. Adv. Manuf. Technol.*, vol. 81, no. 1, pp. 693–703, 2015.

[14] B. Naderi and R. Ruiz, “The distributed permutation flowshop scheduling problem,” *Comput. Oper. Res.*, vol. 37, no. 4, pp. 754–768, 2010.

[15] Y. Unlu and S. J. Mason, “Evaluation of mixed integer programming formulations for non-preemptive parallel machine scheduling problems,” *Comput. Ind. Eng.*, vol. 58, no. 4, pp. 785–800, 2010.

[16] L. Meng, C. Zhang, X. Shao, B. Zhang, Y. Ren, and W. Lin, “More MILP models for hybrid flow shop scheduling problem and its extended problems,” *Int. J. Prod. Res.*, vol. 58, no. 13, pp. 3905–3930, 2020.

[17] C. H. Pan, “A study of integer programming formulations for scheduling problems,” *Int. J. Syst. Sci.*, vol. 28, no. 1, pp. 33–41, 1997.

[18] B. Naderi, S. Gohari, and M. Yazdani, “Hybrid flexible flowshop problems: Models and solution methods,” *Appl. Math. Model.*, vol. 38, no. 24, pp. 5767–5780, 2014.

[19] L. Meng, C. Zhang, Y. Ren, B. Zhang, and C. Lv, “Mixed-integer linear programming and constraint programming formulations for solving distributed flexible job shop scheduling problem,” *Comput. Ind. Eng.*, vol. 142, p. 106347, 2020.

[20] P. C. Gilmore and R. E. Gomory, “Multistage cutting stock problems of two and more dimensions,” *Oper. Res.*, vol. 13, no. 1, pp. 94–120, 1965.

[21] R. Macedo, C. Alves, and J. V. De Carvalho, “Arc-flow model for the two-dimensional guillotine cutting stock problem,” *Comput. Oper. Res.*, vol. 37, no. 6, pp. 991–1001, 2010.

[22] M. Mrad, I. Meftahi, and M. Haouari, “A branch-and-price algorithm for the two-stage guillotine cutting stock problem,” *J. Oper. Res. Soc.*, vol. 64, no. 5, pp. 629–637, 2013.

[23] R. Alvarez-Valdes, A. Parajon, and J. M. Tamarit, “A computational study of LP-based heuristic algorithms for two-dimensional guillotine cutting stock problems,” *OR Spectr.*, vol. 24, no. 2, pp. 179–192, 2002.

[24] F. Furini, E. Malaguti, R. M. Durán, A. Persiani, and P. Toth, “A column generation heuristic for the two-dimensional two-staged guillotine cutting stock problem with multiple stock size,” *Eur. J. Oper. Res.*, vol. 218, no. 1, pp. 251–260, 2012.

[25] K. Tieng, S. Sumetthapiwat, A. Dumrongsiri, and C. Jeenanunta, “Heuristics for two-dimensional rectangular guillotine cutting stock,” *Thail. Stat.*, vol. 14, no. 2, pp. 147–164, 2016.

[26] S. Sumetthapiwat, B. Intiyot, and C. Jeenanunta, “A column generation on two-dimensional cutting stock problem with fixed-size usable leftover and multiple stock sizes,” *Int. J. Logist. Manag.*, vol. 35, no. 2, pp. 273–288, 2020.

[27] W. Wei, L. Jian-Yong, and W. Heng, “Path optimization for PCB NC-drilling using genetic algorithm,” *CEA.*, vol. 44, no. 229, 2008.

[28] M. S. Saealal, A. F. Abidin, A. Adam, J. Mukred, K. Khalil, Z. Yusof, Z. Ibrahim, and N. Nordin, “An ant colony system for routing in PCB holes drilling process,” *IJIMIP*, vol. 3, no. 1, pp. 50–56, 2012.

[29] G. Onwubolu and M. Clerc, “Optimal path for automated drilling operations by a new heuristic approach using particle swarm optimization,” *Int. J. Prod. Res.*, vol. 42, no. 3, pp. 473–491, 2004.

[30] W. C. E. Lim, G. Kanagaraj, and S. Ponnambalam, “Cuckoo search algorithm for optimization of sequence in PCB holes drilling process,” in *Emerging trends in science, engineering and technology*, pp. 207–216, Springer, 2012.

[31] W. C. E. Lim, G. Kanagaraj, and S. Ponnambalam, “PCB drill path optimization by combinatorial cuckoo search algorithm,” *Sci. World J.*, vol. 2014, 2014.

[32] G. Kanagaraj, S. Ponnambalam, and W. C. E. Lim, “Application of a hybridized cuckoo search-genetic algorithm to path optimization for PCB holes drilling process,” in *2014 IEEE International Conference on Automation Science and Engineering (CASE)*, pp. 373–378, IEEE, 2014.

[33] P. Ji, M. Sze, and W. B. Lee, “A genetic algorithm of determining cycle time for printed circuit board assembly lines,” *Eur. J. Oper. Res.*, vol. 128, no. 1, pp. 175–184, 2001.

[34] D. M. Kodek and M. Krisper, “Optimal algorithm for minimizing production cycle time of a printed circuit board assembly line,” *Int. J. Prod. Res.*, vol. 42, no. 23, pp. 5031–5048, 2004.

[35] S. Emet, T. Knuutila, E. Alhoniemi, M. Maier, M. Johnsson, and O. S. Nevalainen, “Workload balancing in printed circuit board assembly,” *Int. J. Adv. Manuf. Technol.*, vol. 50, no. 9-12, pp. 1175–1182, 2010.

[36] T. He, D. Li, and S. W. Yoon, “A heuristic algorithm to balance workloads of high-speed SMT machines in a PCB assembly line,” *Procedia Manuf.*, vol. 11, pp. 1790–1797, 2017.

[37] P. Damodaran, K. Srihari, and S. S. Lam, “Scheduling a capacitated batch-processing machine to minimize makespan,” *Robot. Comput. Integr. Manuf.*, vol. 23, no. 2, pp. 208–216, 2007.

[38] P. Damodaran, D. A. Diyadawagamage, O. Ghrayeb, and M. C. Vélez-Gallego, “A particle swarm optimization algorithm for minimizing makespan of nonidenti-

cal parallel batch processing machines," *Int. J. Adv. Manuf. Technol.*, vol. 58, no. 9, pp. 1131–1140, 2012.

[39] P. Damodaran and K. Srihari, "Mixed integer formulation to minimize makespan in a flow shop with batch processing machines," *Math. Comput. Model.*, vol. 40, no. 13, pp. 1465–1472, 2004.

[40] M. Hulett, P. Damodaran, and M. Amouie, "Scheduling non-identical parallel batch processing machines to minimize total weighted tardiness using particle swarm optimization," *Comput. Ind. Eng.*, vol. 113, pp. 425–436, 2017.

[41] J.-S. Chen and J.-S. Yang, "Model formulations for the machine scheduling problem with limited waiting time constraints," *J. Inf. Optim. Sci.*, vol. 27, no. 1, pp. 225–240, 2006.

[42] B. Naderi, S. Fatemi Ghomi, M. Aminnayeri, and M. Zandieh, "A study on open shop scheduling to minimise total tardiness," *Int. J. Prod. Res.*, vol. 49, no. 15, pp. 4657–4678, 2011.

[43] M. Mousakhani, "Sequence-dependent setup time flexible job shop scheduling problem to minimise total tardiness," *Int. J. Prod. Res.*, vol. 51, no. 12, pp. 3476–3487, 2013.

[44] M. Saidi-Mehrabad and P. Fattahi, "Flexible job shop scheduling with tabu search algorithms," *Int. J. Adv. Manuf. Technol.*, vol. 32, no. 5, pp. 563–570, 2007.

[45] P. Fattahi, M. S. Mehrabad, and F. Jolai, "Mathematical modeling and heuristic approaches to flexible job shop scheduling problems," *J. Intell. Manuf.*, vol. 18, no. 3, pp. 331–342, 2007.

[46] V. Roshanaei, H. ElMaraghy, and A. Azab, "Sequence-based MILP modeling for flexible job shop scheduling," in *IIE Annual Conference Proceedings*, p. 1, Institute of Industrial and Systems Engineers (IISE), 2012.

[47] B. Naderi and A. Azab, "Modeling and heuristics for scheduling of distributed job shops," *Exp. Syst. Appl.*, vol. 41, no. 17, pp. 7754–7763, 2014.

[48] F. Wang, Q. Tang, Y. Rao, C. Zhang, and L. Zhang, “Efficient estimation of distribution for flexible hybrid flow shop scheduling,” *Acta. Autom. Sin.*, vol. 43, no. 2, pp. 280–293, 2017.

[49] M. Pinedo, *Scheduling*, vol. 29. Springer, 2012.

APPENDICES

APPENDIX A : IBM ILOG OPL CPLEX code for Model 1.

This section demonstrates the CPLEX code for Model 1, which includes .mod file and .dat file as follows.

```
%.mod file
/*** Parameters ***/
int I = 5;
range panels = 1..I;
int K = 6;
range stainlesses = 1..K;
int L = 8;
range layouts = 1..L;
int P = 6;
range pressmachines = 1..P;
int O = 3;
range ovens = 1..O;
int T = 12;
range cycles = 1..T;
range cycles1 = 1..(T-1);
int m = 10;
int n = 120;
int M = 10000;

/*** Import the values of a(ikl) from excel file ***/
int temp[1..I*K*L] = ...;
int a[i in panels, k in stainlesses, l in layouts]= temp[l+L*(k-1)+K*L
    *(i-1)];
int d[panels] = [500, 500, 500, 500, 500];

/*** Decision Variables ***/
dvar boolean x[panels][stainlesses][layouts][pressmachines][cycles];
dvar boolean X[pressmachines][cycles][ovens];
dvar boolean Y[pressmachines][cycles][pressmachines][cycles][ovens];
dvar float+ A[pressmachines][cycles];
dvar float+ B[pressmachines][cycles][ovens];
dvar float+ C[pressmachines][cycles];
dvar float+ D[pressmachines][cycles][ovens];
dvar float+ Ch[pressmachines][cycles]; %Ch is C'
dvar float+ Cmax;

/*** Objective function ***/
minimize Cmax;

/*** Constraints ***/
subject to {
   forall(p in pressmachines, t in cycles)
      sum(i in panels, k in stainlesses, l in layouts) x[i][k][l][p][t
         ] <= 1;

   forall(i in panels, k in stainlesses, l in layouts, p in
       pressmachines, t in cycles)
      x[i][k][l][p][t] <= a[i][k][l];

   forall(i in panels)
      sum(k in stainlesses, l in layouts, p in pressmachines, t in
         cycles)x[i][k][l][p][t]*(m*a[i][k][l]) >= d[i];

   forall(p in pressmachines, t in cycles1)
      sum(i in panels, k in stainlesses, l in layouts)x[i][k][l][p][t]
          >= sum(i in panels, k in stainlesses, l in layouts)x[i][k][
         l][p][t+1];

   forall(p in pressmachines, t in cycles)
      sum(o in ovens) X[p][t][o] == 1;

```

```
forall(p in pressmachines, t in cycles, o in ovens)
    B[p][t][o]+D[p][t][o] <= X[p][t][o]*M;

forall(p in pressmachines, t in cycles1)
    A[p][t+1] >= C[p][t];

forall(p in pressmachines, t in cycles)
    sum(o in ovens) B[p][t][o] == A[p][t]+n;

forall(p in pressmachines, t in cycles)
    C[p][t] == A[p][t]+3*n;

forall(p in pressmachines, t in cycles, o in ovens)
    D[p][t][o] >= B[p][t][o]+n-(1-X[p][t][o])*M;

forall(p in pressmachines, t in cycles, o in ovens)
    D[p][t][o] <= B[p][t][o]+n+(1-X[p][t][o])*M;

forall(p in pressmachines, t in cycles, pp in pressmachines:pp!=p,
    tt in cycles, o in ovens)
    B[p][t][o] >= D[pp][tt][o]-Y[p][t][pp][tt][o]*M;

forall(p in pressmachines, t in cycles, pp in pressmachines:pp!=p,
    tt in cycles, o in ovens)
    B[pp][tt][o] >= D[p][t][o]-(1-Y[p][t][pp][tt][o])*M;

forall(p in pressmachines, t in cycles)
    C[p][t]-M*(1- sum(i in panels, k in stainlesses, l in layouts)x[
        i][k][l][p][t] ) <= Ch[p][t];

forall(p in pressmachines, t in cycles)
    Ch[p][t] <= C[p][t]+M*(1- sum(i in panels, k in stainlesses, l
        in layouts)x[i][k][l][p][t] );

    forall(p in pressmachines, t in cycles)
        Ch[p][t] <= M*(sum(i in panels, k in stainlesses, l in layouts)x
            [i][k][l][p][t]);

    forall(p in pressmachines, t in cycles)
        Cmax >= Ch[p][t];
}

/*** The following is written to report some results ***/
execute {
    % Report the output of each panel type
    writeln("Total products");
    for (var i in panels) {
        var Totali = 0;
        for (var k in stainlesses){
            for (var l in layouts){
                for (var p in pressmachines){
                    for (var t in cycles){
                        Totali = Totali+x[i][k][l][p][t]*(m*a[i][k][l])
                    }
                }
            }
        }
        writeln("Total products of panel",i," are ", Totali);
    }
    writeln(" ");

    % Report the variable x(iklpt) that is equal to 1
    writeln("x[i][k][l][p][t] that is equal to 1 sort according to p,t")
        ;
    for (var p in pressmachines){
        for (var t in cycles){
            for (var i in panels) {
                for (var k in stainlesses){
                    for (var l in layouts){
                        if (x[i][k][l][p][t] == 1){
                            writeln("x[",i,"][",k,"][",l,"][",p,"][",t,"]=
                                ",x[i][k][l][p][t]);
                            writeln("a[",i,"][",k,"][",l,"]= ",a[i][k][l]);
                        }
                    }
                }
            }
        }
    }
}
```

```
%.dat file
SheetConnection sheetInput("aikl5panel.xls");
temp from SheetRead(sheetInput,"Sheet1!A1:A240");
```

APPENDIX B : IBM ILOG OPL CPLEX code for Model 2.

This section demonstrates the CPLEX code for Model 2, which includes .mod file and .dat file as follows.

```
%.mod file
/*** Parameters ***/
int I = 5;
range panels = 1..I;
int K = 6;
range stainlesses = 1..K;
int L = 8;
range layouts = 1..L;
int P = 6;
range pressmachines = 1..P;
int O = 3;
range ovens = 1..O;
int T = 12;
range cycles = 1..T;
range cycles1 = 1..(T-1);
int m = 10;
int n = 120;
int M = 5000;

/*** Import the values of a(ikl) from excel file ***/
int temp[1..I*K*L] = ...;
int a[i in panels, k in stainlesses, l in layouts]= temp[l+L*(k-1)+K*L
    *(i-1)];
int d[panels] = [500, 500, 500, 500, 500];

/*** Decision Variables ***/
dvar boolean x[panels][stainlesses][layouts][pressmachines][cycles];
dvar boolean X[pressmachines][cycles][ovens];
dvar boolean Y[pressmachines][cycles][pressmachines][cycles];
dvar float+ A[pressmachines][cycles];
dvar float+ B[pressmachines][cycles];
dvar float+ Ah[pressmachines][cycles]; %Ah is A'
dvar float+ Cmax;

/*** Objective function ***/
minimize Cmax;

/*** Constraints ***/
subject to {
   forall(p in pressmachines, t in cycles)
      sum(i in panels, k in stainlesses, l in layouts) x[i][k][l][p][t
          ] <= 1;

   forall(i in panels, k in stainlesses, l in layouts, p in
       pressmachines, t in cycles)
      x[i][k][l][p][t] <= a[i][k][l];

   forall(i in panels)
      sum(k in stainlesses, l in layouts, p in pressmachines, t in
          cycles)x[i][k][l][p][t]*(m*a[i][k][l]) >= d[i];

   forall(p in pressmachines, t in cycles1)
      sum(i in panels, k in stainlesses, l in layouts)x[i][k][l][p][t]
           >= sum(i in panels, k in stainlesses, l in layouts)x[i][k][
          l][p][t+1];

   forall(p in pressmachines, t in cycles)
      sum(o in ovens) X[p][t][o] == 1;

   forall(p in pressmachines, t in cycles1)
      A[p][t+1] >= A[p][t]+3*n;
```

```

    forall(p in pressmachines, t in cycles)
        B[p][t] == A[p][t]+n;

    forall(pp in pressmachines, tt in cycles, p in pressmachines:p<pp, t
         in cycles, o in ovens)
        B[pp][tt] >= B[p][t]+n-(3-Y[p][t][pp][tt]-X[p][t][o]-X[pp][tt][o
            ])*M;

    forall(pp in pressmachines, tt in cycles, p in pressmachines:p<pp, t
         in cycles, o in ovens)
        B[p][t] >= B[pp][tt]+n-(2+Y[p][t][pp][tt]-X[p][t][o]-X[pp][tt][o
            ])*M;

    forall(p in pressmachines, t in cycles)
        A[p][t]-M*(1- sum(i in panels, k in stainlesses, l in layouts)x[
            i][k][l][p][t] ) <= Ah[p][t];

    forall(p in pressmachines, t in cycles)
        Ah[p][t] <= A[p][t]+M*(1- sum(i in panels, k in stainlesses, l
            in layouts)x[i][k][l][p][t]);

    forall(p in pressmachines, t in cycles)
        Ah[p][t] <= M*(sum(i in panels, k in stainlesses, l in layouts)x
            [i][k][l][p][t]);

    forall(p in pressmachines, t in cycles)
        Cmax >= Ah[p][t]+3*n;
}

/*** The following is written to report some results ***/
execute {
    % Report the output of each panel type
```

```
writeln("Total products");
for (var i in panels) {
    var Totali = 0;
    for (var k in stainlesses){
        for (var l in layouts){
            for (var p in pressmachines){
                for (var t in cycles){
                    Totali = Totali+x[i][k][l][p][t]*(m*a[i][k][l])
                }
            }
        }
    }
    writeln("Total products of panel",i," are ", Totali);
}
writeln(" ");

% Report the variable x(iklpt) that is equal to 1
writeln("x[i][k][l][p][t] that is equal to 1 sort according to p,t")
    ;
for (var p in pressmachines){
    for (var t in cycles){
        for (var i in panels) {
            for (var k in stainlesses){
                for (var l in layouts){
                    if (x[i][k][l][p][t] == 1){
                        writeln("x[",i,"][",k,"][",l,"][",p,"][",t,"]=
                            ",x[i][k][l][p][t]);
                        writeln("a[",i,"][",k,"][",l,"]= ",a[i][k][l]);
                    }
                }
            }
        }
    }
    }
}
```

```
%.dat file
SheetConnection sheetInput("aikl5panel.xls");
temp from SheetRead(sheetInput,"Sheet1!A1:A240");
```

APPENDIX C : Python code for the 3P-PCB-PH.

The following code is used to find the list *aikl*. This list collects the matrices that correspond to panel types, and each matrix contains the number of panels of type $i \in \hat{I}$ per opening using SST size $k \in \hat{K}$ and layout $l \in \hat{L}$. The list *aikl* will be used as an input in the 3P-PCB-PH algorithm.

```
import numpy as np
I = 7
K = 6
L = 8
panel_data = np.array([[20.5, 24, 0.5, 0.25], #warp, fill, inner gap,
    outer gap
                    [25.65, 22.25, 1, 0.5],
                    [26, 24, 0.5, 0.25],
                    [26.5, 22.5, 1, 0.5],
                    [19, 22.25, 0.5, 0.25],
                    [15, 23.8, 0.5, 0.25],
                    [27.75, 20.5, 0.5, 0.25]])
stainless_data = np.array([[50, 44], #warp, fill
                        [50, 53],
                        [50, 56],
                        [50, 58],
                        [43, 25.5],
                        [43, 27]])
aikl = []

for i in range(I):
  temp = np.zeros((K,L))
  a = paneldata[i][0]
  b = paneldata[i][1]
  g = paneldata[i][2]
  G = paneldata[i][3]
  for k in range(K):
    X = stainless[k][0]
    Y = stainless[k][1]
    temp[k][0] = int( np.floor((X-2*(G-g/2))/(a+g)) * np.floor((Y-2*(G-g/2))/(b+g)) )
    temp[k][1] = int( np.floor((X-2*(G-g/2))/(b+g)) * np.floor((Y-2*(G-g/2))/(a+g)) )
    temp[k][2] = int( np.floor((X-2*(G-g/2))/(a+g)) + ( np.floor((X-2*(G-g/2))/(b+g)) * np.floor((Y-b-G-2*(G-g/2))/(a+g)) ) )
    temp[k][3] = int( np.floor((Y-2*(G-g/2))/(a+g)) + ( np.floor((Y-2*(G-g/2))/(b+g)) * np.floor((X-b-G-2*(G-g/2))/(a+g)) ) )
    temp[k][4] = int( np.floor((X-2*(G-g/2))/(b+g)) + ( np.floor((X-2*(G-g/2))/(a+g)) * np.floor((Y-a-G-2*(G-g/2))/(b+g)) ) )
    temp[k][5] = int( np.floor((Y-2*(G-g/2))/(b+g)) + ( np.floor((Y-2*(G-g/2))/(a+g)) * np.floor((X-a-G-2*(G-g/2))/(b+g)) ) )
    temp[k][6] = int( np.floor((X-2*(G-g/2))/(a+g)) )
    temp[k][7] = int( np.floor((X-2*(G-g/2))/(b+g)) )
  aikl.append(temp)
```

The following is the code for the 3P-PCB-PH algorithm.

```
import time
import numpy as np
import statistics
list_collect_time = []

#Input
I = 7
K = 6
L = 8
P = 6
O = 3
T = 16
```

```
m = 10
n = 120
di = [500, 420, 595, 375, 330, 680, 580]

for z in range(10):
  time_start = time.time()

  ############# Phase1 #############
  xikl = []   #collect [i,kbar,lbar,aiklbar] of each panel type i
  for i in range(I):
    aiklbar = np.amax(aikl[i])
    position = np.where(aikl[i] == aiklbar)
    listOfCordinates = list(zip(position[0], position[1]))
    xikl.append([i, listOfCordinates[0][0], listOfCordinates[0][1],
        aiklbar])

  list_dci = []
  for i in range(I):
    list_dci.append(np.ceil(di[i]/(m*xikl[i][3])))
  dc = int(sum(list_dci))
  #print('number of cycles that satisfies demand =', list_dci)
  #print("dc =", dc)

  ############# Phase 2 #############
  A = np.zeros((P,T))
  C = np.zeros((P,T))
  Can = []
  for i in range(P):
    Can.append(0)
  Oven_Scedule_List = []
  for i in range(O):
    Oven_Scedule_List.append([])

```

```
for j in range(dc):
  #Choose press machine p_p
  p_p = Can.index(min(Can))

  #Choose oven o_p
  temp2 = []
  for i in range(O):
    temp2.append(len(Oven_Scedule_List[i]))
  o_p = temp2.index(min(temp2))
  if Can[p_p] == 0:
    start_time_press_machine = 0
  else:
    start_time_press_machine = C[p_p][Can[p_p]-1]
  if len(Oven_Scedule_List[o_p]) == 0:
    A[p_p][Can[p_p]] = start_time_press_machine
    C[p_p][Can[p_p]] = start_time_press_machine + 3*n
    start_time_oven = start_time_press_machine + n
    end_time_oven = start_time_oven + n
    temp3 = [start_time_oven, end_time_oven, p_p, Can[p_p]]
    Oven_Scedule_List[o_p].append(temp3)
    Can[p_p] = Can[p_p]+1
  else:
    # find idle_time (Oven_Idle_Time_List)
    sorted(Oven_Scedule_List[o_p], key = lambda x: x[0])
    idle_time = []
    if len(Oven_Scedule_List[o_p]) == 1:
      idle_time.append([0,Oven_Scedule_List[o_p][0][0]])
      idle_time.append([Oven_Scedule_List[o_p][0][1], float('inf')])
    else:
      for i in range(len(Oven_Scedule_List[o_p])):
        if i == 0:
          if Oven_Scedule_List[o_p][i][0] != 0:
            idle_time.append([0,Oven_Scedule_List[o_p][i][0]])
```

```
   elif i == len(Oven_Scedule_List[o_p])-1:
    if Oven_Scedule_List[o_p][i-1][1] != Oven_Scedule_List[o_p][i][0]:
     idle_time.append([Oven_Scedule_List[o_p][i-1][1], Oven_Scedule_List[o_p][i][0]])
    idle_time.append([Oven_Scedule_List[o_p][i][1], float('inf')])
   else:
    if Oven_Scedule_List[o_p][i-1][1] != Oven_Scedule_List[o_p][i][0]:
     idle_time.append([Oven_Scedule_List[o_p][i-1][1], Oven_Scedule_List[o_p][i][0]])

for i in range(len(idle_time)): # for else
 if idle_time[i][0] > start_time_press_machine+n:
  star = i
  case = 0
  break
else:
 case = 1
 start_time_oven = start_time_press_machine+n
 end_time_oven = start_time_oven+n
 A[p_p][Can[p_p]] = start_time_oven-n
 C[p_p][Can[p_p]] = end_time_oven+n
 temp3 = [start_time_oven, end_time_oven, p_p, Can[p_p]]
 Oven_Scedule_List[o_p].append(temp3)
 Can[p_p] = Can[p_p]+1

if case == 0:
 for i in range(len(idle_time)):
  if i < star:
   if (start_time_press_machine+n)+n <= idle_time[i][1]:
    start_time_oven = start_time_press_machine+n
```

```
            end_time_oven = start_time_oven+n
            A[p_p][Can[p_p]] = start_time_oven-n
            C[p_p][Can[p_p]] = end_time_oven+n
            temp3 = [start_time_oven, end_time_oven, p_p, Can[p_p]]
            Oven_Scedule_List[o_p].append(temp3)
            Can[p_p] = Can[p_p]+1
            break
        else:
          if idle_time[i][0]+n <= idle_time[i][1]:
            start_time_oven = idle_time[i][0]
            end_time_oven = start_time_oven+n
            A[p_p][Can[p_p]] = start_time_oven-n
            C[p_p][Can[p_p]] = end_time_oven+n
            temp3 = [start_time_oven, end_time_oven, p_p, Can[p_p]]
            Oven_Scedule_List[o_p].append(temp3)
            Can[p_p] = Can[p_p]+1
            break

############# Phase 3 #############
xiklpt = []
t = 0
p = 0
for i in range(I):
  for j in range(int(list_dci[i])):
    xiklpt.append([xikl[i][0], xikl[i][1], xikl[i][2], p ,t])
    p = p+1
    if p == P:
      p = 0
      t = t+1

############# Print Output #############
finished_goods = []
for i in range(I):
    temp4 = m*xikl[i][3]*list_dci[i]
    finished_goods.append(temp4)
  print('finished_goods = ', finished_goods)

  print('A =',A)
  print('C =',C)

  print('xiklpt = ',xiklpt)

  for i in range(len(Oven_Scedule_List)):
    print('Oven_Scedule_List[',i,']=',Oven_Scedule_List[i])

  print('Cmax = ', C.max())

  print("This is end of", z, "iteration")
  time_end = time.time()
  print('time of this iteration',z,'is ', time_end-time_start)
  list_collect_time.append(time_end-time_start)
  print("************************************************************")

print(list_collect_time)
print('mean of run time is ', statistics.mean(list_collect_time))
print('SD of run time is ', statistics.stdev(list_collect_time))
```

BIOGRAPHY

Name	Mr. Teeradech Laisupannawong
Date of Birth	November 23, 1993
Place of Birth	Ratchaburi, Thailand
Educations	B.S. (Mathematics) (First Class Honours), Kasetsart University, 2016 M.Sc. (Applied Mathematics and Computational Science), Chulalongkorn University, 2018

Publications

- Laisupannawong, T., Intiyot, B., & Jeenanunta, C. (2021). Mixed-Integer Linear Programming Model and Heuristic for Short-Term Scheduling of Pressing Process in Multi-Layer Printed Circuit Board Manufacturing. *Mathematics*, 9(6), 653.
- Laisupannawong, T., Intiyot, B., & Jeenanunta, C. (2021). Improved Mixed-Integer Linear Programming Model for Short-Term Scheduling of the Pressing Process in Multi-Layer Printed Circuit Board Manufacturing. *Mathematics*, 9(21), 2653.