國立交通大學

工業工程與管理學系

博士論文

液晶顯示模組裝配廠生產排程問題之研究

The Study on the Production Scheduling Problems for Liquid Crystal Display Module Assembly factories

學生：戴于婷

指導教授：鍾淑馨博士

中華民國九十七年七月

液晶顯示模組裝配廠生產排程問題之研究

The Study on the Production Scheduling Problems for
Liquid Crystal Display Module Assembly factories

研究生：戴于婷　　Student: Yu-Ting Tai

指導教授：鍾淑馨 博士　　Advisor: Dr. Shu-Hsing Chung

國 立 交 通 大 學
工業工程與管理學系
博士論文

A Dissertation Submitted to
Department of Industrial Engineering and Management
College of Management
National Chiao Tung University
in Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy
in
Industrial Engineering and Management

July 2008
Hsinchu, Taiwan, Republic of China

中華民國九十七年七月

液晶顯示模組裝配廠生產排程問題之研究

學生：戴于婷 *指導教授：鍾淑馨博士*

國立交通大學工業工程與管理系

摘　　要

在液晶面板廠中，如何充分利用產能及滿足顧客交期來增加競爭力及獲利，為一重要之研究課題；在液晶面板製造的最終模組裝配製造階段中，印刷電路板焊接作業為生產過程中的瓶頸，其排程結果將影響整個模組裝配廠的生產績效，因此，本論文先針對此瓶頸作業所衍生的平行機台排程問題進行探討。而在經過印刷電路板焊接作業後，工件在出貨前還須經過老化測試作業，這是模組裝配廠中的批次機台，老化測試的排程問題是一個多維度的排程問題，其考量了不同的到臨時間、不同的工件大小、有限的機台產能以及批次相依的加工時間，不當的老化測試排程將影響出貨時程，因此，發展老化測試排程的求解程序在模組裝配廠也是一重要的課題。

本論文首先解決印刷電路板焊接排程問題 (printed circuit board bonding scheduling problem；PCBSP)，其為順序相依設置時間的平行機台排程問題，工件在此問題中分屬不同的產品族，由於模組裝配廠所製造的產品面臨契約與現貨並存的混合市場，因此，工件在 PCBSP 中依據利潤及客戶重要性等因素被賦予不同的權重，在此問題中，本文於滿足契約數量及不違反交期及機器產能的限制下，以求得最大化總加權產出作為 PCBSP 的求解目標，這些工件一旦於瓶頸資源上完工，則需要加速出貨，因此，在老化測試排程問題 (aging test scheduling problem；ATSP) 中則以最小化完工時間為目標。

在本論文中，我們針對 PCBSP 與 ATSP 兩個排程問題分別建立其混合整數規劃模式，亦將 PCBSP 的子問題轉換為考量時間窗限制下之車輛路線問題(vehicle routing problem with time window；VRPTW)，並修改既有的網路演算法來發展新的啟發式解法，同時也應用貪婪法則來解決其他的子問題；此外，針對 ATSP，我們也設計一個複合啟發式法則來決定批次個數，並將此個數視為一已知的參數帶入混合整數規劃模式中，以降低原先模式求解的複雜度，而針對大型的 ATSP，本文也提出三個啟發式解法來求解；經由運算結果及績效比較中得知，本論文針對 PCBSP 及 ATSP 所提出的啟發式法則有很好的求解效果。

關鍵詞：平行機台排程、批次、加權產出、順序相依設置時間、模組裝配

The Study on the Production Scheduling Problems for Liquid-Crystal-Display Module Assembly Factories

Student：Yu-Ting Tai　　Advisor： Dr. Shu-Hsing Chung

ABSTRACT

To be competitive, the thin film transistor liquid crystal display (TFT-LCD) companies need to utilize their capacity and to satisfy customers' due dates in order to increase their profitability. In the final stage of TFT-LCD, module assembly manufacturing process, the printed circuit board (PCB) bonding usually causes bottleneck in production; its schedule mainly affect the system performance of module assembly factories. Therefore, an essential scheduling problem is tackled for the bottleneck operation first; it is referred to as the printed circuit board bonding scheduling problem (PCBSP). Furthermore, following the PCB bonding operation, there exists an aging test operation, which is the only batch server in the whole process. The aging test scheduling problem (ATSP) is complicated because it is a multi-dimensional problem, which involves the constraints of unequal ready times, non-identical job sizes, limited machine capacity, and batch dependent processing times. Therefore, the development of efficient algorithms is also critical to form appropriate batches and to arrange a suitable schedule for those jobs which have been processed by the PCB bonding operation.

For the PCBSP, the jobs are clustered by their product types, which must be processed on identical parallel machines. They are also given various weights based on their profits and customer importance due to hybrid contracted and spot markets being in the module assembly industry. Furthermore, setup times for two consecutive jobs between different product types on the same machines are sequence-dependent. Thus, the objective for the PCBSP is to maximize the total weighted throughput subject to fulfilling contracted quantities without violating the due dates and machine capacity restrictions. Once those jobs are planned and processed in the bottleneck operation, they

should be accelerated their speed to complete and delivery. Consequently, the ATSP with a minimal makespan criterion is also considered.

In this dissertation, the PCBSP and ATSP are formulated as two mixed integer linear programming models. For the PCBSP, it can be viewed as a multi-level optimization problem, the sub-problem of PCBSP can be transformed into the vehicle routing problem with time window (VRPTW), a well-known network routing problem which has been investigated extensively. We present two new algorithms based on the network algorithms with some modifications to accommodate the PCBSP. Furthermore, the greedy concept is also applied to the other sub-problem of PCBSP. For the ATSP, an effective compound algorithm is proposed to determine the number of batches and to apply this number as one parameter in the MILP model in order to reduce the complexity of the problem. Three efficient heuristic algorithms are also provided for the large-scaled ATSP. Computational results and performance comparisons show that the proposed algorithms, which are used to solve the PCBSP and ATSP, are efficient and near-optimal.

Keywords : parallel machines scheduling, batch, weighted throughput, sequence dependent setup time, module assembly.

誌　　謝

這本論文能順利地完成，要感謝的人真的太多了，都是因為大家的幫忙，我才能一步一步地走完這個旅程。首先，要感謝我的指導老師 鍾淑馨教授， 鍾老師治學嚴謹，做研究及教學都相當認真，對學生生活上的關懷更是無微不至，每每在我學業、家庭、身體與工作產生衝突時，鍾老師總是給我滿滿的鼓勵與建議，讓我能在這條路上，將自己的角色扮演好，做好每一件事，能有幸認識老師，並在研究的方法與態度上獲得老師的啟發與影響真是我的福氣，在此向恩師致上崇高的敬意與謝意。另外，也要感謝 彭文理教授，彭老師總是不厭其煩地回答我的問題，有老師的鼓勵，也讓我有更多跌倒了再爬起來的勇氣，此外，口試期間，承蒙口試委員 張百棧院長、吳泰熙教授、陳正芳教授及彭文理教授給予的寶貴建議，使本論文更加完備，在此致上深深的謝意。

博士班修業期間，感謝 519 與 517 實驗室的學長姐與學弟妹的鼓勵與幫助，也感謝建瑋同學，不僅在修業期間給我鼓勵，畢業後若有回新竹總是不忘看看老同學，為我加油，也感謝春美學姊與俊穎，常提供給我寶貴的建議，讓我能學習更多，使我的論文更完善，還有雅靜學妹，謝謝妳常常在我最無助的時候，給我一個親切的微笑，讓我能繼續走下去，謝謝你們。

最後，要感謝我的爸爸及親愛的家人，爸爸從小總是對我有著深深的期許，真的很高興自己做到了，沒有辜負他對我的期望，也感謝媽媽以及辛苦的公公婆婆，謝謝你們在我唸書時幫我帶我兩個小孩，讓我能在新竹專心學業，無後顧之憂，當然，也要感謝我的先生 建穎，還有我那兩個可愛的孩子，謝謝建穎總是幽默地在我身旁幫我加油打氣，協助我渡過每一個困難，另外，也謝謝兩位小孩天真無邪的笑容常帶給我莫大的鼓勵。在此向每位關心過我、鼓勵過我的人致上最深的謝意，也將這份成果獻給你們。

戴于婷　僅誌于國立交通大學
管理學院工業工程與管理學系
中華民國九十七年七月十一日

Table of Contents

List of Figures

List of Tables

Notation

Notations for the printed circuit board bonding scheduling problem:

Indices:

i : product type index, $i = 0,1,\ldots,I$;

j : index of job for product type index i, $j = 0,1,\ldots,J_i$;

k : machine index, $k = 1,2,\ldots,K$;

m_k : the k^{th} machine;

H : the set of jobs to be processed;

M : the set of machine containing identical parallel machines;

H_i : job cluster containing J_i jobs of product type i to be processed;

h_{ij} : index for jobs in cluster H_i ;

Parameters:

K : the number of identical machines;

I : the number of job clusters in job set H ;

J_i : the number of jobs in job cluster H_i ;

n_{ij} : lot size of job h_{ij} ;

p_i : the unit processing time for job h_{ij} ;

w_{ij} : the nonnegative weight for job h_{ij} ;

d_{ij} : the due date for job h_{ij} ;

$s_{ii'}$: the sequence dependent setup time between any two consecutive jobs;

Cap : the predetermined machine capacity expressed in terms of time;

$J_i^{contract}$: the number of contract jobs in job cluster H_i ;

e_{ij} : the latest starting time to process job h_{ij} ;

r_{ij} : the ready time for job h_{ij} ;

$SA_{ii'}$: the savings for the pairs of two jobs associated with product type H_i *and* $H_{i'}$;

$PSA_{h_{ij}h_{i'j'}}$: the savings applied in the parallel savings algorithm for pairs of jobs h_{ij} and $h_{i'j'}$;

$GSA_{h_{ij}h_{i'j'}}$: the savings applied in the generalized savings algorithm for pairs of jobs h_{ij} and $h_{i'j'}$;

α_1 : the parameter represents weight of savings for calculating $PSA_{h_{ij}h_{i'j'}}$;

β_1 : the parameter represents weight of weighted throughput ratio for calculating $PSA_{h_{ij}h_{i'j'}}$;

r_1 : the parameter represents weight of time windows restrictions for calculating $PSA_{h_{ij}h_{i'j'}}$;

α_2 : the parameter represents weight of savings for calculating $GSA_{h_{ij}h_{i'j'}}$;

β_2 : the parameter represents weight of weighted throughput ratio for calculating $GSA_{h_{ij}h_{i'j'}}$;

r_2 : the parameter represents weight of time windows restrictions for calculating $GSA_{h_{ij}h_{i'j'}}$;

Decision variables:

x_{ijk} : the variable indicating whether a specific job is scheduled on a machine, with $x_{ijk}=1$ if job h_{ij} is scheduled on machine m_k , and $x_{ijk}=0$ otherwise;

$y_{iji'j'k}$: the precedence variable defined on two jobs h_{ij} and $h_{i'j'}$ scheduled on machine m_k , with $y_{iji'j'k}=1$ if job h_{ij} precede job $h_{i'j'}$ (not necessarily directly), and $y_{iji'j'k}=0$ otherwise;

$z_{iji'j'k}$: the direct-precedence variable defined on two jobs h_{ij} and $h_{i'j'}$

scheduled on machine m_k, with $z_{iji'j'k}=1$ if job h_{ij} direct precede job $h_{i'j'}$, and $z_{iji'j'k}=0$ otherwise.

t_{ijk} : the starting time for job h_{ij} processed on machine m_k;

Notations for the aging test scheduling problem:

Indices:

j : job index, $j=0,1,\ldots,N$;

b : batch index, $b=1,2,\ldots,B$;

k : machine index, $k=1,2,\ldots,K$;

m_k : the k^{th} machine;

Parameters:

N : the number of jobs;

B : the number of batches;

K : the number of machines;

p_j : the processing time for job j;

r_j : the ready time for job j;

s_j : the job size for job j;

S' : the maximum number of pieces can be processed simultaneously on a machine;

μ : a constant, which is chosen to be a sufficiently small value which cannot affect the makespan in Model P;

Q_1 : a constant and is greater than the total number of jobs (N) in Model P;

Q_2 : a chosen constant as it is sufficiently large in value to satisfy $y_{bb'k}=0$ or 1 in Model P;

α : the parameter is used to examine whether there exists a job might be combined with the delayed batch to avoid delaying that job;

β : the parameter represents to accommodate the postponement idea of the DELAY heuristic algorithm ;

Decision variables:

x_{jbk} : the variable indicating whether a specific job is assigned to batch b and scheduled on a machine m_k, with $x_{jbk}=1$ if job j is assigned to batch b and scheduled on machine m_k, and $x_{jbk}=0$ otherwise;

$y_{bb'k}$: the precedence variable defined on two batches b and b' scheduled on machine m_k, with $y_{bb'k}=1$ if batch b precede job b' (not necessarily directly), and $y_{bb'k}=0$ otherwise;

z_{bk} : the variable indicating whether a specific batch b is scheduled on a machine m_k, with $z_{bk}=1$ if batch b is scheduled on machine m_k, and $z_{bk}=0$ otherwise;

t_{bk} : the starting time of batch b to be processed on machine;

$C_{\max}$: the makespan;

pt_b : the longest processing time of all the jobs processed simultaneously in the b^{th} batch;

$C'_{\max}$: the lower bound of makespan in Model N;

BN : the number of lower bound batches obtained from Model N;

Z : the solution obtained from Model P;

Z^* : the solution of the compound MILP-based algorithm obtained within the limited computational time;

t : the decision time point;

r'_b : the batch ready times;

T_b : the summation of the batch ready time and batch processing time.

1. Introduction

1.1. Motivation

In recent years, applications of thin film transistor liquid crystal display (TFT-LCD) products have been increasing rapidly, for example, cellular phones, computer monitors, and LCD TVs. Not surprisingly, TFT-LCD manufacturing has attracted much attention. The TFT-LCD manufacture process consists of four major stages: TFT array fabrication, color filter (CF) fabrication, LCD assembly and module assembly, as depicted in Figure 1-1.

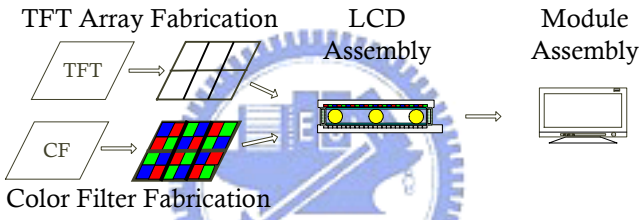


Figure 1-1 The TFT LCD process flow.

TFT array and color filter fabrications are similar to the semiconductor wafer fabrication, and their process steps are also characterized by re-entrant flow. The LCD assembly simultaneously attaches the TFT and color filter and fills the gap between them with liquid crystal. The final stage, module assembly, generally consists of two main segments, front end and back end. It contains the following six processes steps in which the customer-specified components are assembled into the cells: (1) the COG (chip on glass) / TAB (tape automated bonding) process, (2) the attachment of the flexible printed circuit (FPC) board, (3) the bonding of the printed circuit board (PCB), (4) the assembly of the backlight, (5) the aging test, and (6) the inspection, as shown in Figure 1-2.

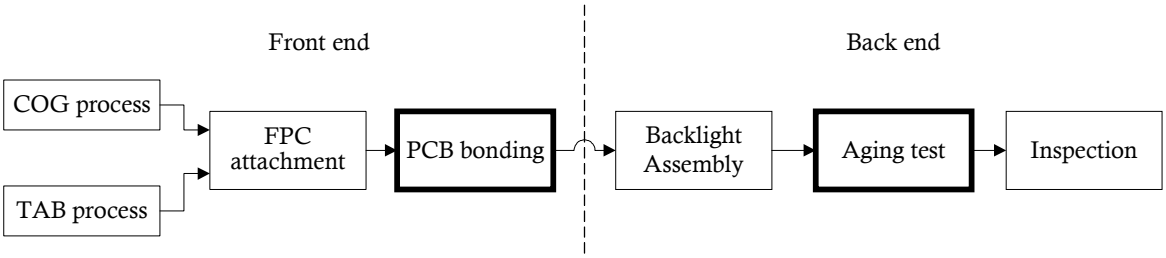


Figure 1-2 The six steps in the module assembly process.

In the module assembly stage, the printed circuit board (PCB) bonding operation usually causes bottleneck in production because it involves the most expensive equipment and its machine utilization is the highest in this stage. Setup times for two consecutive jobs of different product types on the same PCB bonding machine are long and sequentially dependent. Furthermore, in the module assembly manufacturing industry, there exists a hybrid market involving contract and spot markets. The contract jobs associated with contracted prices and due dates have been placed and booked an amount of capacity. Moreover, the other remaining capacity is to be sold into the spot market. The further spot prices are uncertain and high price volatility due to today's fierce competitive environment. In this dissertation, jobs in module assembly factories are given different weights, which are determined by the factors such as job's profit and customer importance. According to the essential concept of theory of constraints (TOC) [41], the performance of a system is determined by the bottleneck in the system. Thus, in this dissertation, a printed circuit board bonding scheduling problem (PCBSP) is first investigated and which selects a appropriate subset of jobs to maximize weighted throughput by considering sequence dependent setup times, job weights, and due dates.

After jobs are processed at the processing of the bottleneck operation (PCB bonding), they should be passed through two main operations, backlight assembly and

aging test (see Figure 1-2) in the module assembly process. In the backlight assembly operation, jobs can be processed smoothly because the non-critical operation has enough capacity and it is the serial server that is the same fashion as the PCB bonding operation. However, how to schedule the jobs processed at the aging test operation is much more complicated since this operation is a batch server which involvs parallel batch machines. Batches at the aging test operation need to be formed by collecting unequal ready-time jobs and non-identical size jobs with considerations of limited machine capacity (a maximum number of pieces can be processed simultaneously on a machine) and batch dependent ready times and processing times. Therefore, solving such multi-dimensional batch processing machines scheduling problem may cause a large makespan if an improper batch-formation strategy is applied. Moreover, Mönch *et al.* [62] also pointed out "*it is sometimes advantageous to process a non-full batch to avoid excessive delays in waiting for jobs with later ready times*". Therefore, an elaborate solution procedure for the aging testing scheduling problem (ATSP) located at the end of module assembly process is necessary. In this dissertation, we not only focus on the investigation for the bottleneck operation by solving the PCBSP, but also expedite the jobs at the aging test operation to complete with a minimal makespan criterion in order to subordinate the scheduling results of the bottleneck operation. Consequently, it is essential that the development of efficient algorithms to solve the ATSP in module assembly factories.

Since the PCBSP and ATSP involve many real-world constraints, they are more difficult to solve than the classical parallel-machine scheduling problems considered by So [78], Shin and Leon [77], and Jeong *et al.* [47] and parallel-batch-processing machine scheduling problems considered by Lee and Uzsoy [51], and Chang *et al.* [14],

respectively. We believe that the development of the scheduling methods for the two scheduling problems can assist those involved in module assembly factories to make judicious scheduling decisions.

1.2. Research Scope and Objectives

In the dissertation, we first focus on the scheduling problem on the bottleneck resource in module assembly process factories; it is referred to as printed circuit board bonding scheduling problem (PCBSP). The PCBSP determines an appropriate subset of jobs to be processed and schedules jobs sequences with a weighted throughput criterion. Subsequently, we consider the batch scheduling problem for the aging test operation which locates at the end of module assembly processing steps. The investigation on the aging test scheduling problem (ATSP) with a minimal makespan criterion provides a schedule for those jobs, which have scheduled in the PCBSP.

The purposes of this dissertation are to develop exact solutions and efficient algorithms for the practical module assembly scheduling problems based on the existing network and batch formation technologies. Two research works will be accomplished in this dissertation:

1. With consideration of the characteristics of different job weights, sequence dependent setup times, due dates, product type dependent processing time, and machine capacity, a mixed integer linear programming model for the PCBSP is constructed to obtain the exact solution. Then, two three-phase algorithms, which based on the existing network algorithms and the greedy concept, are presented to solve the PCBSP efficiently.

2. With the considerations of unequal ready times, non-identical job sizes, limited machine capacity, and batch dependent processing times, a mixed integer linear programming is presented. In addition, an effective compound algorithm and three efficient heuristic algorithms are also provided.

1.3. Organization of the Dissertation

This dissertation focuses on developing the scheduling algorithms for module assembly manufacturing industries is organized as follows.

Chapter 1 provides the motivation of the research and defines the research domain and its objectives.

Chapter 2 presents a literature in the areas involving the existing production management problems in TFT-LCD factories, the scheduling problems with a weighted throughput criterion, network problems, and batch processing machine scheduling problems.

Chapter 3 considers the printed circuit board bonding scheduling problem (PCBSP) in the module assembly to maximize the weighted throughput subject to fulfilling contracted quantities without violating the due date and machine capacity restrictions. The PCBSP is formulated as a mixed integer linear programming model. Two modified algorithms are also proposed to solve the PCBSP efficiently.

Chapter 4 considers the aging test scheduling problem (ATSP) in the module assembly to minimize the makespan. The ATSP is formulated as a mixed integer linear programming model. An effective compound algorithm is proposed to determine the number of batches which is then used as one parameter in the MILP

model in order to reduce the complexity of the problem. Three efficient heuristic algorithms for solving the large-scale parallel batch processing machine scheduling problem are also provided.

Chapter 5 provides the conclusions and several further extensions. Conclusions will be drawn based on the results of the research.

2. Literature Review

The topic of the scheduling problems investigated in this dissertation for module assembly factories bases on ideas from four different research areas. Therefore, this literature review is divided into four areas which contribute to this dissertation regarding related existing production management problems in TFT-LCD factories, the scheduling problems with a weighted throughput criterion, essential network problems, and batch processing machine scheduling problem.

2.1. Existing Production Management Problems in TFT-LCD factories

The amount of available literature which investigates related production management problems in TFT-LCD factories is limited. As can be seen in Table 2-1, those research papers can be classified into three categories according their research scopes: supply chain planning problems [48][56][57][90], post-mapping yield problems [18][80][81][92], and scheduling problems [45][47][50][53][77].

In first category, since four basic process stages are involved in the TFT-LCD industry, the supply chain planning problems in TFT-LCD have been receiving attention from academic researchers. Jeong *et al*. [48] have developed an available-to-promise (ATP) system for TFT-LCD manufacturing in global supply chain environment. The ATP system was used to estimate the promising delivery date for new orders and calculate the unused production capacity with given production schedules. They have also proposed a heuristic for scheduling TFT-LCD module assembly process using the unused capacity at the shop floor level. Lin and Chen [56] and Wang *et al*. [90] have provided a monolithic model of the multi-stage and

Table 2-1 The literature related to production management problems in TFT-LCD factories.

Year	Authors and Refs.	Research scope	Performance criterion	Methodology
2001	Jeong *et al*. [47]	Scheduling problem	Minimize flow time and to maximize the fulfillment of production demands	Linear programming and heuristics
2002	Jeong *et al*. [48]	Available-to–promise system	Generate the more optimistic delivery date promise	Heuristics
2003	Lee and Lee [53]	Production control policies for re-entrant process	Maximal throughput; Minimize the difference of current and target WIP levels	Linear programming
2003	Hu [45]	Scheduling problem	Minimize inventory and backlog	Heuristics
2004	Shin and Leon [77]	Scheduling problem	Minimize total tardiness and the number of family setups	Heuristics
2005	Lai [50]	Scheduling problem	Minimize total tardiness	Heuristics
2005	Su *et al.* [80]	Post-mapping yield problem	Yield improvement	Linear programming (Hungarian method) and heuristic
2005	Chao [18]	Post-mapping yield problem	Minimize total cost of losses and backorder	Linear programming
2006	Su and Yang [81]	Post-mapping yield problem	Yield improvement	Genetic algorithm and simulated annealing
2006	Wang and Su [92]	Yield mapping problem	Maximize the yield rate	Linear programming (Hungarian method) and heuristics (random and greedy)
2007	Lin and Chen [56]	Supply network planning problem	Minimize total costs	MILP
2007	Lin *et al.* [57]	Product mix	Maximize contribution margin	Three-phase mathematical methodology
2007	Wang *et al*. [90]	Product mix	Maximize net profit	MILP

multi-site production planning problem and a MILP model to construct a product mix in TFT-LCD factories, respectively. Lin *et al.* [57] considered a capacity and product mix planning problem for TFT array multi-plant with a maximum contribution margin. They proposed a three-phase methodology which involves capacity configuration, capacity expansion, and capacity exploitation phases. However, in this dissertation, we devote to solving the scheduling problem which the long and middle-term production planning and product mix are given.

The second category, post-mapping yield problem has found by Su *et al.* [80][81], and Wang and Su [92]. They have solved the problem using a series of tools which involve linear programming, heuristics, and meta-heuristics (genetic algorithm and simulated annealing). They revealed that proposed approaches which can solve practical problems effectively and improve the yield rate in the LCD assembly process. Moreover, Chao [18] also provided a liner programming model in order to solve the mapping problem regarding the mapping between TFT and CF substrates.

Finally, the third category is the scheduling problems in TFT-LCD industries. Jeong *et al.* [47] presented mathematical models and proposed two heuristic algorithms to minimize flow time and to maximize the fulfillment of production demands in LCD assembly processes. Lee and Lee [53] have suggested three kinds of control policies: push, push-pull, and pull types. They have used the linear programming models to evaluate performances of the three kinds of control policies. It is also presented in their investigation that a pull type control policy gives stable throughput and delivery satisfaction at a small cost and with less production. Shin and Leon [77] discussed the liquid crystal display module stage scheduling problem. They

provided two heuristics based on the MULTI-FIT method and tabu search to minimize total tardiness and number of family setups. Hu [45] and Lai [50] proposed heuristic algorithms for the lot sizing scheduling problem in color filter factories. Hu [45] and Lai [50] solved their scheduling problems with criterion of minimizing the cost of inventory and backlog and minimizing tardiness, respectively. However, none of these investigations considered sequence dependent setup time and job weights simultaneously. This dissertation investigates the weighted throughput criterion of a module assembly factory which, till now, has been ignored.

2.2. Scheduling Problems with a Maximal Throughput Criterion

During the last decade, there have been many researchers who have investigated the identical parallel machine scheduling problem which is dependent on the completion time of all jobs. The objectives usually involve (see Cheng and Sin [19] for a comprehensive survey) completion time-based [61], due-date based [6][76], and flow-time based [24][43] performance measures, etc. However, So [78] points out that determining the best schedule to process all the jobs currently on hand is not practical. Instead, he suggests choosing one subset of jobs for which to construct daily work schedules according to the existing capacity and demand. However, the PCBSP we investigated in this dissertation select the appropriate job sets to maximize the weighted throughput by considering setup times and job weights.

There are relatively few papers, as can be seen in Table 2-2, which addressed the scheduling problem with a throughput criterion. Hwang and Chang [46] and Tovia *et al.* [85] have focused on the semiconductor manufacturing process and assumed that setup times are negligible. Hwang and Chang [46] have designed a lagrangian

relaxation-based hierarchical production scheduling engines to maximize total number of weighted moves in semiconductor manufacturing. Tovia *et al.* [85] have presented a mathematical programming model and a rule-based heuristic approach to

Table 2-2 The literature for the scheduling problems with a throughput maximization criterion.

Authors and Refs.	Setup time	Preemption	Performance criterion (max.)	Shop type
Hwang and Chang [46]	negligible	non-preemptive	total weighted moves	Job shop
Tovia *et al.* [85]	negligible	non-preemptive	throughput	parallel machines
Baptiste *et al.* [4]	negligible	preemptive	total weighted throughput	single machine
Fung *et al.* [34]	negligible	preemptive	total weight	single machine
So [78]	sequence independent batch setup time	non-preemptive	total reward	parallel machines
Hiraishi *et al.* [44]	sequence independent setup time	non-preemptive	weighted number of just-in-time jobs	parallel machines
Rojanasoonthon *et al.* [74]	sequence dependent setup time	non-preemptive	weighted number of scheduled jobs	parallel machines

solve a throughput maximization problem in a semiconductor packaging factory. Furthermore, Baptiste *et al.* [4] and Fung *et al.* [34] have considered the preemptive scheduling problem with equal length processing and negligible setup time. However, setup times and non-preemptive scheduling characteristics of the front-end module assembly manufacturing process are essential and significant.

Allahverdi *et al.* [1] conducted an extensively survey for the scheduling problems which involve setup times. They pointed out that So [78] is the only one who has considered a total reward objective for parallel machine scheduling problems prior to

1999. So [78] considered an analogous version of the PCBSP and presented three heuristics to solve the problem approximately. He tackled a problem that existed in a minor setup time between jobs of the same family and a major setup time between jobs from different groups. According to the classification presented by Allahverdi *et al.* [1], what So [78] considered classified to the sequence independent batch setup times, which is the special case of sequence dependent setup time. Since 1999, with respect to the total weight of completed jobs, Hiraishi *et al.* [44] and Rojanasoonthon *et al.* [74] have maximized the weighted number of just-in-time jobs and the weighted number of scheduled jobs, respectively, as solutions to the scheduling problems they investigated. However, Hiraishi *et al.* [44] in their consideration of JIT job and Rojanasoonthon *et al.* [74] in their examination of strict order of priority in the job completion schedule, both limit and reduce the throughput of parallel machines.

2.3. Network Problems

Two classical network problems, the traveling salesman problem (TSP) [58] and the vehicle routing problem (VRP) [7][79], are among the most widely investigations combinatorial optimization problems. The TSP can be stated as follows: a salesman, starting in one city, wishes to visit other cities once and exactly once and return to the start with minimal total distance traveled. Furthermore, the VRP involves the design of a set of minimum-cost vehicle routes without violating the limited vehicle capacity, originating and terminating at a central depot, for a fleet of vehicles that serviced a set of customers with known demand [7][79]. There are many generalizations of the well-known TSP and VRP. Examples include the traveling salesman problem with time windows (TSPTW) [28][35], the traveling salesman problem with profits [32], the traveling salesman's subtour problem [36], the prize-collecting traveling salesman

problem [3], the prize-collecting traveling salesman problem with time windows (TW-TSP) [5], the vehicle routing problem with time windows (VRPTW) [79], the orienteering problem (OP) [11][16][39][40], the multiple tour maximum collection problem (MTMCP) [10][11], and the team orienteering problem (TOP) [2][17][84].

It should be noted that jobs in PCBSP are taken the constraint of due dates into consideration and are processed on parallel machines. To solve the scheduling problem for PCBSP and arrange jobs' sequence, this dissertation adopts the basic technologies of vehicle routing problem with time windows (VRPTW). Hence, we further focus on the VRPTW in detail. The VRPTW is a well-known network routing problem which has been investigated extensively [8][9][79]. Bräysy and Gendreau [8] given a explicit definition regarding the VRPTW as follows: "*The VRPTW can be described as the problem of designing least cost routes from one depot to a set of geographically scattered points. The routes must be designed in such a way that each point is visited only once by exactly one vehicle within a given time interval, all routes start and end at the depot, and the total demands of all points on one particular route must not exceed the capacity of the vehicle*." The VRPTW is NP-complete which is demonstrated by Lenstra and Rinnooy Kan [54]. A large amount of solution procedures, involving route-construction methods, route-improvement methods, and meta-heuristics, have been applied to the VRPTW. Bräysy and Gendreau [8][9] have provided complete surveys of research papers regarding the VRPTW.

VRPTW can be used to solve the scheduling problems in semiconductor manufacturing factories, such as the wafer probing factories [65][68] and IC final testing factories [69]. Pearn *et al.* [65][68] have transformed the wafer probing

scheduling problem (WPSP) into the vehicle routing problem with time windows (VRPTW). They gave an illustrative example to demonstrate the proposed transformation and used the existing VRPTW algorithms to solve the WPSP near-optimally. They also developed three modified algorithms to solve the WPSP. In the IC final testing factories, Pearn *et al.* [69] applied the network algorithms designed for the VRPTW to solve the IC final testing problem with the characteristic of reentry based on the stage-by-stage solution strategy and full-load policy applied to batch-processing stages, so that the capacity of critical resources testers would be efficiently utilized. In this dissertation, the author took the merits of the VRPTW and applied the VRPTW algorithms to the scheduling problems with different job weights, sequence dependence setup times, due dates in module assembly factories.

2.4. Batch Processing Machine Scheduling Problems

In this dissertation, the aging test scheduling problem (ATSP) considers a parallel batch processing machine scheduling problem on the aging test operation in the module assembly process. In recent years, much research has focused on providing solutions to the batch processing machine (BPM) scheduling problems on a single or parallel batch processing machines. A comprehensive literature and a classification scheme on batch processing machine scheduling problems are presented by Potts and Kovalyov [73]. For the batch processing machine scheduling problems in semiconductor manufacturing, Mathirajan and Sivakumar [59] have provided a complete survey.

There are two types of batch processing machine scheduling problem, which are named as incompatible product family and compatible problem family. The former is

that jobs with different product families are mutually incompatible for processing in the same batch; the latter is assumed that jobs belonging to different product families may be simultaneously processed. In this dissertation, the aging test scheduling problem we investigated considers the batch processing of compatible product families. In problems of this type, the batch processing time is computed by the longest job processing time in that batch.

The literature regarding the BPM scheduling problems on a single or parallel batch processing machines with compatible product families is shown in Table 2-3. As presented in Table 2-3, the first researchers to address the batch processing scheduling problem arising in a burn-in oven of the final test in the semiconductor industry are Lee *et al.* [52]. They used dynamic programming-based algorithms and heuristics for a number of performance measures, such as maximum tardiness ($T_{\max}$), the number of tardy jobs ($\sum U_i$), and maximum lateness time ($L_{\max}$) on a single batch processing machine. They have also presented heuristics for the parallel batch processing machine scheduling problem with the minimum makespan ($C_{\max}$) and maximum lateness time ($L_{\max}$) criteria. They have explored the area of scheduling batch processing machines and offered a classification of complexity for the investigated problems. Furthermore, Chandru *et al.* [12][13], DuPont and Ghazvini [30], Poon and Yu [71], Mönch *et al.* [63] provided the solutions for the single/parallel batch processing machine scheduling problems with identical job sizes. Although the single batch processing machine scheduling problem (Uzsoy [86], Ghazvini and DuPont [37], Zhang *et al.* [94], DuPont and Dhaenens- Flipo [29], Melouk *et al.* [60], Erramilli

Table 2-3 The literature related to the batch processing machine scheduling problem with compatible product families.

Year	Authors and Refs.	Shop type	Ready time	Job size	Performance criterion
1992	Lee *et al.* [52]	single machine/ parallel machines	unequal/ equal	identical/ identical	$T_{\max}$, $\sum U_i$, $L_{\max}$ / $C_{\max}$, $L_{\max}$
1993	Chandru *et al.* [12]	single machine	equal	identical	$C_{\max}$
1993	Chandru *et al.* [13]	single machine/ parallel machines	equal	identical	$C_{\max}$
1997	DuPont and Ghazvini [30]	single machine	equal	identical	$\overline{F}_i$
2004	Poon and Yu [71]	single machine	equal	identical	$\sum C_i$
2006	Mönch *et al.* [63]	single machine	equal	identical	$\sum \lvert d_i - C_i \rvert$
1994	Uzsoy [86]	single machine	equal	non-identical	$C_{\max}$, $\sum C_i$
1998	Ghazvini and DuPont [37]	single machine	equal	non-identical	$\overline{F}_i$
2001	Zhang *et al.* [94]	single machine	equal	non-identical	$C_{\max}$
2002	DuPont, and Dhaenens-Flipo [29]	single machine	equal	non-identical	$C_{\max}$
2004	Melouk *et al.* [60]	single machine	equal	non-identical	$C_{\max}$
2004	Damodaran and Srihari [25]	Two machines in a flow shop	equal	non-identical	$C_{\max}$
2006	Erramilli and Mason [31]	single machine	equal	non-identical	$\sum w_i T_i$
2006	Kashan *et al.* [49]	single machine	equal	non-identical	$C_{\max}$
1999	Lee and Uzsoy [51]	single machine	unequal	identical	$C_{\max}$
2000	Sung and Choung [82]	single machine	equal/ unequal	identical	$C_{\max}$
2002	Sung *et al.* [83]	single machine	unequal	identical	$C_{\max}$
2002	Wang and Uzsoy [89]	single machine	unequal	identical	$L_{\max}$
2004	Li and Lee [55]	single machine	unequal	identical	$T_{\max}$, $\sum U_i$
2004	Poon and Zhang [70]	single machine	unequal	identical	$C_{\max}$
2004	Deng *et al.* [26]	single machine	unequal	identical	$\sum w_i C_i$
2005	Deng *et al.* [27]	single machine	unequal	identical	$\sum C_i$
2005	Poon and Yu [72]	single machine	unequal	identical	$C_{\max}$
2006	Gupta and Sivakumar [42]	single machine	unequal	identical	$T_{\max}$, $\sum U_i$, $\sum T_i / n$
2004	Van Der Zee [87]	single machine	dynamic arrival	identical	$\overline{F}_i$
2004	Chang and Wang [15]	single machine	unequal	non-identical	$\sum C_i$
2006	Chou *et al.* [21]	single machine	unequal	non-identical	$C_{\max}$
2007	Chou [20]	single machine	unequal	non-identical	$C_{\max}$
2007	Wang *et al.* [91]	single machine	unequal	non-identical	$C_{\max}$
2007	Chou and Wang [22]	single machine	unequal	non-identical	$\sum w_i T_i$
2004	Chang *et al.* [14]	parallel machines	equal	non-identical	$C_{\max}$
2007	Mönch and Unbehaun [64]	parallel machines	equal	identical	$\sum \lvert d_i - C_i \rvert$
2007	Van Der Zee [88]	parallel machines	dynamic arrival	non-identical	$\overline{F}_i$
2008	ATSP (this dissertation)	parallel machines	unequal	non-identical	$C_{\max}$

and Mason [31], Kashan *et al.* [49]) and the two batch processing machines in a flow shop (Damodaran and Srihari [25]) took the non-identical job sizes into consideration to reflect more practical situations, they have assumed that the ready times of jobs for batch processing machines are equal. This assumption prevents the developed procedures from being directly applied to the parallel batch processing machine scheduling problem investigated in this dissertation because the ATSP involves unequal ready times.

For the single batch processing machine scheduling problem with non-identical job sizes and equal ready times, Uzsoy [86] investigated this type problem to minimize the total completion times ($\sum C_i$) of the jobs and makespan. He has also provided bin-packing-based heuristics for minimizing makespan and has used the branch and bound approach to minimize the total completion times. He also developed effective heuristics for the criteria of minimum makespan and minimum total completion time. Erramilli and Mason [31] have investigated the multiple orders per job (MOJ) problem on a single batch processing machine. They grouped different customer orders into jobs and combined jobs into batches and scheduled them on a single batch processing machine to minimize the total weighted tardiness ($\sum w_i T_i$) of orders. Damodaran and Srihari [25] have proposed two mathematical models with the minimum makespan criterion to schedule batches of jobs on two machines in a flow shop. Kashan *et al.* [49] has addressed the need to minimize makespan by employing two different genetic algorithms (GAs) for scheduling jobs with non-identical sizes on a single batch processing machine. Unfortunately, all the above

models do not consider the unequal ready time that is a common phenomenon in module assembly factories.

Although Lee and Uzsoy [51], Sung and Choung [82], Sung *et al.* [83], Wang and Uzsoy [89], Li and Lee [55], Poon and Zhang [70], Deng *et al.* [26], [27], Poon and Yu [72], Gupta and Sivakumar [42], and Van Der Zee [87] have considered the characteristic of unequal ready times, they limited their applications to a single batch processing machine and an identical job size. Lee and Uzsoy [51] have provided efficient heuristics to solve the scheduling problem arising in the final test phase of semiconductor manufacturing. To minimize the maximum completion time on a single batch processing machine with dynamic job arrivals, they designed three algorithms (GRLPT, DELAY, and UPDATE) to find the approximate solutions. Sung and Choung [82] have presented a branch-and-bound algorithm and several heuristics to solve the static and dynamic cases on a single batch processing machine. Their objective was also to minimize the makespan of all jobs. Sung *et al.* [83] have considered a single batch processing machine with job families and dynamic job arrivals. The performance measure used to evaluate a schedule is the minimum makespan. Van Der Zee [87] has also presented the dynamic control of a batch processing machine; his objective was to find the minimum average flow time per product in the presence of compatible product families.

Moreover, in recent years, a series of research papers regarding single batch processing machine scheduling problem with unequal ready times and non-identical job size are provided by Chang, Chou, and Wang [15][20][21][22][91]. First, Chang and Wang [15] investigated the single machine problem of scheduling semiconductor

burn-in operation with non-identical job sizes and unequal ready time. They provided an efficient heuristic algorithm and examine the effect of arrival time and processing time on minimizing the total completion time. Subsequently, Chou *et al.* [21] changed the objective into minimal makespan and proposed a merge-split procedure to improve the solution obtained by the longest processing time batch first fit (LPT-BFF). Furthermore, two hybrid genetic algorithms (GA) were also provided. Following that, Chou [20] presented a solution procedure to joint GA and DP (dynamic programming). Wang *et al.* [91] provided a mix integer programming model to describe problem complexity. Simultaneously, a hybrid forward/backward algorithm is also presented and the computational results showed good performances of this algorithm in term of solution quality within a modest computational time. Moreover, Chou and Wang [22] took the distinct due date into consideration and investigated the single machine scheduling problem with a minimal total weighted tardiness criterion. They proposed one MIP model and two hybrid heuristics involving a rule-based, GA, and DP algorithms. The computational results indicated GA-based algorithm outperformed the rule-based algorithm in terms of solution quality for small size problems.

More recently, the parallel batch processing machine scheduling problem with compatible product family characteristic is considered by Mönch and Unbehaun [64], Chang *et al.* [14], and Van Der Zee [88]. Mönch and Unbehaun [64] presented a parallel batch processing machine scheduling problem in which jobs have identical job sizes and equal ready times. The objective is to minimize the sum of the absolute deviations of completion times from the due date of all jobs. They proposed three heuristics based on exact algorithm, genetic algorithm, dynamic programming

techniques. Chang *et al.* [14] have provided a mathematical model and developed an algorithm based on simulated annealing (SA) approach to minimize makespan for the scheduling problem with equal ready times and non-identical job size. They have not included the unequal ready times in their model. Van Der Zee [88] extended the scheduling problem [87] involved single machine to parallel machines. The objective of the parallel batch scheduling problem is to minimize average flow time per product. He developed a new look-ahead strategy to solve this problem. However, the processing time in his model is assumed fixed. Their solution procedure cannot be applied to ATSP directly.

At the time this dissertation was being written, the author was not aware of any other studies of the parallel batch processing machine scheduling problem with unequal ready time, non-identical job size, and compatible product family characteristics. Therefore, this dissertation arises from the need in industry to consider jobs with these practical situations, which are processed on identical parallel batch processing machines.

3. Algorithms for the Printed Circuit Board Bonding Scheduling Problem

In this chapter, we address the first scheduling problem, the printed circuit board bonding scheduling problem (PCBSP), which is a practical variant of the classical parallel machine scheduling problem. The objective for the PCBSP is to maximize the total weighted throughput subject to fulfilling contracted quantities without violating the due dates and machine capacity restrictions. In this chapter, we present two heuristic procedures to solve the PCBSP efficiently based on some basic technologies used for developing algorithms for machine scheduling. Furthermore, computational results and performance comparisons are also provided.

3.1. Introduction

Printed circuit board bonding is the bottleneck resource in the module assembly process. The performance of the system is determined by this bottleneck operation according to the essential concept of theory of constraints (TOC) [41]. In this chapter, we consider the PCBSP in a module assembly factory with considerations of sequence dependent setup times and multiple job weights. For the PCBSP we investigated, jobs are clustered by their product types, which must be processed on any parallel machines and be completed before the due dates. Setup times between two consecutive jobs of different product types on the same machine are sequentially dependent. The job processing time may vary, depending on the product type of the jobs processed on. Furthermore, the hybrid market consists of contract and spot markets in the TFT-LCD industry. Therefore, jobs are given different weights, which are determined by the factors such as job's profit

and customer importance. Since the PCBSP involved multiple job weights, with constraints on sequence dependence setup times, product type dependent processing times, due dates and machine capacity, it is more difficult to solve than the classical parallel machine scheduling problems.

The PCBSP we investigated can be modeled as a multi-level optimization problem. At the first level, we schedule the contract jobs to minimize the total setup time without violating machine capacity and customer due dates restrictions, which can be solved using algorithms for the vehicle routing problem with time windows (VRPTW) [65][68][69]. At the second level, we apply a greedy concept to choose a subset of spot jobs then insert into the schedules constructed in the first level. In this dissertation, we use basic technologies for VRPTW algorithms including parallel, generalized savings algorithms and provide two modifications to solve the PCBSP efficiently. To further analyze the impact of the problem characteristics on the performance of those savings algorithms, we provide a set of test problems, considering the workload level of contract jobs, tightness of due dates, setup time variation, and variation of contract/spot weight ratio. Exact solutions are used here as convenient reference points for evaluating the accuracy and effectiveness of our heuristic algorithms. The computational experiments and comparisons demonstrate the performance of the three phases of the modified parallel savings algorithm outperform the other algorithms.

3.2. Module Assembly Process and Problem Description

3.2.1. Module Assembly Process

The TFT-LCD applications include monitors, notebook PCs, mobile phones, portable DVDs, LCD TVs, and many others. Such applications are hundreds of product

types of job processed with sizes ranging from 1.6 inches to 46 inches. The manufacturing process of the module assembly generally consists of two main segments, front end and back end. It contains the following six processes, (1) COG (chip on glass) / TAB (tape automated bonding) process, (2) Flexible printed circuit board (FPC) attachment, (3) Printed circuit board (PCB) bonding, (4) Back light assembly, (5) Aging test, (6) Inspection, as presented in Figure 3-1. In the module assembly process, the critical resource is PCB bonding, which has the longest setup time and using anisotropic conductive film (ACF) to connects the PCB and FPC, as illustrated in Figure 3-2.

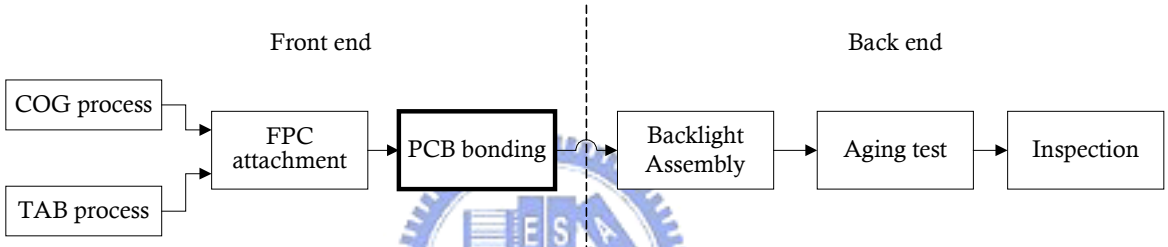


Figure 3-1 The six steps in the module assembly process.

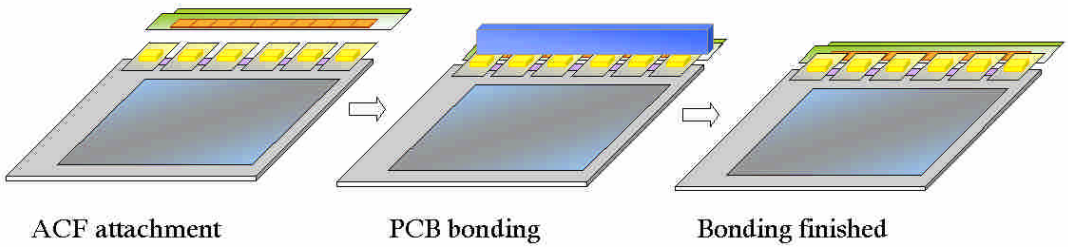


Figure 3-2 PCB bonding illustration.

The processes of PCB bonding is determined by the types of mount technology of LCD drive IC and controller IC, namely, the COG (chip on glass) and TAB (tape automated bonding). The COG is a technology that mounts the LCD driver to the contact edge of the LCD glass, which is depicted in Figure 3-3. The TAB is the LCD driver or controller electronics are encapsulated in a thin film, like package, with metal leads extension from the IC chips, which is depicted as Figure 3-4. The process is also

called OLB (Outer Lead Bonding). In general, large size LCD applications (ranging from 15 inches to 46 inches) adopt the TAB structure and small size LCD applications (ranging from 1.6 inches to 6 inches) adopt the COG structure.

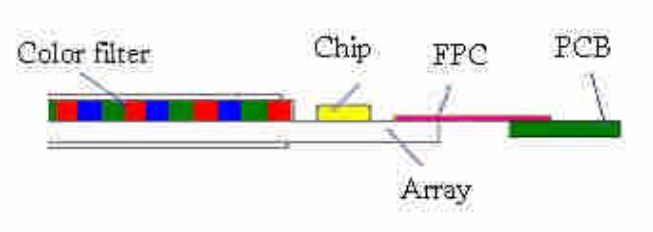


Figure 3-3 COG structure.

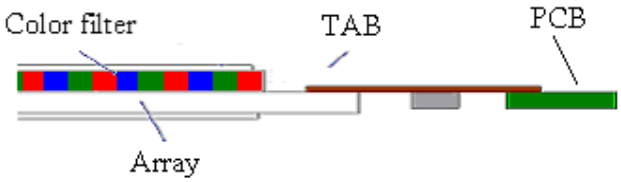


Figure 3-4 TAB structure.

Setup time between different technologies in PCB bonding is complicated and time-consuming. The longest setup time may consume six hours. The setup activities of PCB bonding include the following: cool down temperature, replacement of the appropriate mold, and the rising to a suitable temperature and voltage. Furthermore, setup times between two consecutive jobs of different product types on the same PCB bonding machine are sequentially dependent. In a similar product family, the setup activity may adjust the temperature and voltage. Different LCD panel sizes must replace the mold.

3.2.2. Printed Circuit Board Bonding Scheduling Problem

The scheduling problem in this chapter can be stated as follows. Let machine group $M = \{ m_k \mid k = 1, 2, \ldots, K \}$, contain K machines, and product types $H = \{ H_i \mid i = 1, 2, \ldots, I \}$, contain I clusters of jobs. Each job cluster $H_i = \{ h_{ij} \mid j = 1, 2, \ldots, J_i^{contract}$,

$J_i^{contract}+1, \ldots J_i\}$, representing the job set. Term $J_i^{contract}$ and J_i are the total number of contract jobs and total number of jobs of product type H_i, respectively. Each job in their associated product type is a candidate to be processed without preemption on a set of parallel machines *K*.

Each job h_{ij} carries with its processing time denoted by p_i, and weight denoted by w_{ij}, where $i=\{1,2, \ldots, I\}$ and $j=\{1,2, \ldots, J_i\}$. For each job h_{ij}, r_{ij} represent the ready time of the job to be processed on a machine and e_{ij} represent the latest starting time to process job h_{ij}, which relates to the due dates d_{ij} and can be computed as $e_{ij}=d_{ij}-p_{ij}$. A setup time is incurred in the different product types. When job h_{ij} immediately succeeds job $h_{i'j'}$ on machine m_k, a setup time $s_{ii'}$ happens. The setup time is sequentially dependent on the product types of the jobs processed on the machine. The objective is to maximize the total weighted throughput without violating contracted due date and machine capacity, Cap.

3.2.3. An Illustrative Example

Consider the following example with two machines and seven jobs clustered into three product types. Table 3-1 displays the product type, processing time, job weight, due date, and contracted status for each job. Furthermore, the setup times are incurred for switching one product type to another for the three product types H_{A}, H_{B}, and H_{C}. Table 3-2 shows the required setup times and the term U denotes that the machine is in idle status. The capacity is set to 95 for each machine in the illustrative example.

The objective in the illustrative example is to find a schedule for the subset of jobs, which satisfies the due date restrictions without violating the constraints of machine capacity, while maximizing the total weighted throughput.

Table 3-1 The job information for the 7-job example.

Job ID	Product type	Processing time	Weight	Due date	Contract /Spot
h_{A1}	H_A	21	50	80	C*
h_{A2}	H_A	21	40	100	S**
h_{A3}	H_A	21	40	100	S
h_{B1}	H_B	25	60	93	C
h_{B2}	H_B	25	50	100	S
h_{C1}	H_C	28	63	100	C
h_{C2}	H_C	28	63	60	C

*Term C indicates the contract job. **Term S indicates the spot job.

Table 3-2 Setup times required for switching one product type to another for H_A, H_B, and H_C.

	To			
From	U	H_A	H_B	H_C
U	-	15	15	15
H_A	0	0	10	7
H_B	0	8	0	5
H_C	0	3	16	0

3.3. An Integer Programming Formulation for the PCBSP

In this section, a MILP model is formulated for the PCBSP. The applicability of the MILP model with consideration of sequence dependent setup times for a parallel machine scheduling problem has been demonstrated by Pearn *et al.* [66][67]. The objective function and some constraints of MILP model are modified to accommodate the PCBSP.

Let x_{ijk} be the variable indicating whether job h_{ij} is scheduled on machine m_k, with $x_{ijk}=1$ if job h_{ij} is scheduled to be processed on machine m_k, and $x_{ijk}=0$ otherwise. Let $y_{iji'j'k}$ be the precedence variable, where $y_{iji'j'k}$ should be set to 1 if job $h_{i'j'}$ is scheduled following job h_{ij} (not necessarily directly), and where $y_{iji'j'k}=0$ otherwise. Let $z_{iji'j'k}$ be the direct-precedence variable, with $z_{iji'j'k}=1$ if job $h_{i'j'}$ is scheduled immediately following job h_{ij} on machine m_k, and $z_{iji'j'k}=0$ otherwise.

Further, the starting processing time t_{ijk} should be should not be less than the ready time, and not be greater than the latest starting time e_{ij}. The exact formulation for the PCBSP is as follows.

$$\text{Maximize} \quad Z = \sum_{k=1}^{K}\sum_{i=1}^{I}\sum_{j=1}^{J_i} x_{ijk} w_{ij} \tag{3-1}$$

subject to

$$\sum_{k=1}^{K} x_{ijk} = 1, \quad \text{for} \quad i = 1, 2, \ldots, I, \quad j = 1, 2, \ldots, J_i^{contract} \tag{3-2}$$

$$\sum_{k=1}^{K} x_{ijk} \le 1, \quad \text{for} \quad i = 1, 2, \ldots, I, \quad j = J_i^{contract}+1, J_i^{contract}+2, \ldots, J_i \tag{3-3}$$

Capacity constraints:

$$\sum_{i=0}^{I}\sum_{j=1}^{J_i} x_{ijk} p_i + \sum_{i=0}^{I}\sum_{j=1}^{J_i}\left(\sum_{i'=0}^{I}\sum_{j'=1}^{J_{i'}} z_{iji'j'k}\, s_{ii'}\right) \le Cap, \quad \text{for all } k \tag{3-4}$$

Due date constraints:

$$t_{ijk} \ge r_{ij} x_{ijk}, \quad \text{for all } i, j, k \tag{3-5}$$

$$t_{ijk} \le e_{ij} x_{ijk}, \quad \text{for all } i, j, k \tag{3-6}$$

Precedence constraints:

$$t_{ijk} + p_i + s_{ii'} - t_{i'j'k} + Q(y_{iji'j'k} - 1) \le 0, \quad \text{for all } i, j, k \tag{3-7}$$

$$t_{ijk} + p_i + s_{ii'} - t_{i'j'k} - Q(y_{iji'j'k} + z_{iji'j'k} - 2) \ge 0, \quad \text{for all } i, j, k \tag{3-8}$$

$$(y_{iji'j'k} + y_{i'j'ijk}) - Q(x_{ijk} + x_{i'j'k} - 2) \ge 1, \quad \text{for all } i, j, k \tag{3-9}$$

$$(y_{iji'j'k} + y_{i'j'ijk}) + Q(x_{ijk} + x_{i'j'k} - 2) \le 1, \quad \text{for all } i, j, k \tag{3-10}$$

$$(y_{iji'j'k} + y_{i'j'ijk}) - Q(x_{ijk} + x_{i'j'k}) \le 0, \quad \text{for all } i, j, k \tag{3-11}$$

$$(y_{iji'j'k} + y_{i'j'ijk}) - Q(x_{i'j'k} - x_{ijk} + 1) \le 0, \quad \text{for all } i, j, k \tag{3-12}$$

$$(y_{iji'j'k} + y_{i'j'ijk}) - Q(x_{ijk} - x_{i'j'k} + 1) \le 0, \quad \text{for all } i, j, k \tag{3-13}$$

$$y_{iji'j'k} \ge z_{iji'j'k} \quad \text{for all } i, j, k \tag{3-14}$$

$$\sum_{i=0}^{I}\sum_{j=1}^{J_i} x_{ijk} - \sum_{r_{ij}\neq r_{i'j'}} z_{iji'j'k} = 1, \quad \text{for all } k \tag{3-15}$$

$$y_{iji^*j^*k} + z_{iji^*j^*k} - Q(y_{iji^*j^*k} + z_{iji^*j^*k} - 2) - Q(y_{iji'j'k} - z_{iji'j'k} - 1) \geq 2, \quad \text{for all } i, j, k \tag{3-16}$$

***Binary variables*:**

$$x_{ijk} \in \{0, 1\}, \quad \text{for all } i, j, k \tag{3-17}$$

$$y_{iji'j'k} \in \{0, 1\}, \quad \text{for all } i, j, k \tag{3-18}$$

$$z_{iji'j'k} \in \{0, 1\}. \quad \text{for all } i, j, k \tag{3-19}$$

The objective function (3-1) states that the total weighted throughput is to be maximized over all machines. Constraint (3-2) ensures that each contract job is scheduled on one machine exactly. Constraint (3-3) ensures that each spot job is scheduled at most one machine. Constraint (3-4) is the capacity constraint, which forces the sum of processing time and setup time for each machine within available capacity. Constraints (3-5) and (3-6) state that the starting time t_{ijk} for each job h_{ij} scheduled on machine m_k should not be less than the earlier starting time r_{ij} and not be greater than the latest starting time e_{ij}. Constraints (3-7) and (3-8) are the starting time constraints. The time of the following job starts after the proceeding job and related setup is complete. As usual, Q is a 'sufficiently large' positive number, so that constraints (3-9)-(3-13) are satisfied for $y_{iji'j'k} = 0$ or 1. Constraints (3-9)-(3-13) are the precedence constraints and constraints (3-14)-(3-16) are the direct constraints. These constraints state their sequence relation. In constraint (3-16), variable $z_{iji^*j^*k}$ states that there is a job $h_{i^*j^*}$ existing after job h_{ij} when $y_{iji'j'k} = 1$ and $z_{iji'j'k} = 0$. Constraints (3-17)-(3-19) indicate that x_{ijk}, $y_{iji'j'k}$ and $z_{iji'j'k}$ are binary integer variables. The total number of variables is $2N_I^2K$, and the total number of equations is $N_I^3K - (5/2)N_I^2K - (3/2)N_IK + N_I + 2K$, where $N_I = \sum_{i=1}^{I} J_i$.

3.4. Algorithms for the PCBSP

In this section, two heuristic procedures are presented to solve the PCBSP efficiently. Some basic technologies have been used for developing algorithms for machine scheduling. The conventional savings algorithms are modified to minimize the total setup time for the contract jobs and apply the greedy concept for the spot jobs to enhance the total weighted throughput. We first review two savings algorithms, the parallel savings algorithm proposed by Golden [38] and the generalized savings algorithm provided by Christofides *et al.* [23]. We then present two modified algorithms, which are referred to as the three-phase modified parallel savings algorithm (MPSA_TP) and the three-phase modified generalized savings algorithm (MGSA_TP).

3.4.1. Parallel Savings Algorithm

Golden [38] proposed the parallel savings algorithm (PSA) to solve the TSP approximately. The PSA, initially calculates the savings of all pairs of jobs and sorts those savings in descending order. The PSA creates the multiple of K machines simultaneously at the initial stage, where K is the number of machines. Note that a selected pair of jobs is feasible if it does not violate the machine capacity constraints. The parallel savings algorithm then searches downward from the savings list, for a job which could be merged into one of current K schedules (at the first endpoint or last endpoint) with the most savings. The algorithm terminates when no more jobs can be inserted. Algorithm details are presented as follows.

Step 1. (Initialization) Calculate the savings $SA_{ii'}$, defined as the following, for all pairs of two jobs associated with product type H_i and $H_{i'}$, where U denotes the idle status.

$$SA_{ii'} = s_{Ui} + s_{i'U} - s_{ii'}. \quad (3\text{-}20)$$

Step 2. Sort the savings list in descending order of magnitude.

Step 3. (Schedule initial construction) Choose the first K pairs of jobs on the list satisfying the machine capacity constraints, to start K new schedules simultaneously.

Step 4. Starting from the top of the savings list. Find the first feasible pair on the list to add to one of the two ends of a currently constructed schedule with the most savings.

Step 5. The chosen jobs form a feasible machine schedule. Repeat Step 4 until all schedules are full and cannot be expanded.

3.4.2. Generalized Savings Algorithm

In contrast to the PSA, which constructs a multiple of K machine schedules simultaneously at the initial stage, the generalized savings algorithm (GSA) proposed by Christofides *et al.* [23] creates one schedule at a time and considers not only the end points but also positions between two consecutive jobs when inserting a new job into the current partial schedule. Besides, the insertion costs are calculated for every unscheduled job at every possible position. The chosen job, which maximizes savings while minimizing insertion costs, is used to avoid the algorithm to create a new schedule on another machine with a high setup time.

3.4.3. New Algorithms

To effectively apply these technologies, we modify them to develop fast and effective algorithms. The two new algorithms essentially consist of three phases. Phase I applies

network algorithms to schedule sub-problem of contract jobs with minimum total setup time. Phase II inserts spot jobs near the same constructed product type so as no extra setup time is needed. Phase III sorts the remaining spot jobs with weighted throughput ratio (i.e. job weight is divided by job processing time) and chooses a subset of high weighted throughput ratio jobs which are then inserted into the constructed schedules sequentially until all machine capacities are full. It is noted, however, that the contract jobs in the current constructed schedule should be pushed backward when a spot job is inserted into this schedule in Phase II or Phase III. Thus, the contract jobs, following the inserted spot job, should be re-checked according to their due dates to meet customer deadlines. The new solution procedures can be described as follows.

3.4.3.1. Three-phase Modified Parallel Savings Algorithm

The three-phase modified parallel savings algorithm calculates savings by using the original saving term and two additional terms, weighted throughput ratio and time window restrictions. First, the saving term adopts the basic technology of the saving calculation used for the traveling salesman problem. The term is used to reduce setup times incurred by arranging two jobs which are consecutively processed but belong to different job clusters. Second, the weighted throughput ratio term is added in the savings calculation in order to choose the pair of jobs with higher weighted throughput ratio as the first job pair than the pair of jobs with lower ones. By doing this, the jobs with higher weighted throughput ratios are forced to be processed earlier than other jobs with lower ones. Furthermore, the jobs with the same weighted throughput ratios are forced to be processed closer to each other than the other jobs. Thirdly, the other added term, time window restrictions, takes job h_{ij} whose latest starting time (e_{ij}) is earlier than a job $h_{i'j'}$

with a later starting time ($e_{i'j'}$). The main purpose of the time window restrictions term is that jobs with earlier latest starting times are forced to be processed earlier than the other jobs with later ones.

In the saving calculation of MPSA_TP, three parameters, α_1, β_1, and γ_1, and three the ranges $0 \le \alpha_1 \le 1$, $0 \le \beta_1 \le 1$, and $0 \le \gamma_1 \le 2$, are added to the savings function as the weight of the 'savings term', 'weighted throughput ratio term', and 'time window restrictions term', respectively. Parameter α_1 represents weight of setup time savings, which results from consecutively processing two jobs h_{ij} and $h_{i'j'}$ as a job pair. It can help to avoid a long setup time being incurred. Parameter β_1 is used to weight the sum of the weighted throughput ratios of the two jobs h_{ij} and $h_{i'j'}$ in a pair in order to achieve higher weighted throughput. Finally, parameter γ_1 is used in the time windows restrictions term to prevent the jobs being processed after their due dates in order to enhance customer satisfaction. The time window restriction was designed by Pearn *et al.* [68]; however, they only considered the value $0 \le \gamma_1 \le 1$ and did not systematically examine the parameter with the value $\gamma_1 \ge 1$. Therefore, the available range of γ_1 is enlarged as $0 \le \gamma_1 \le 2$ for investigation. In addition, term $10Cap$ is a scaled number where Cap represents the machine capacity. The scaled number is used to make clear distinction of each saving value because the three proposed terms in the saving calculation have different measurement units. By incorporating the three parameters and the scaled number, the effects of the three terms are enhanced so as to drive appropriate sequencing.

Phase I (Modified savings algorithm)

Step 1. Calculate the savings, $PSA_{h_{ij}h_{i'j'}}$, for all pair of jobs h_{ij} and $h_{i'j'}$, where U

denotes the idle status. $s_{ii'}$ is the required setup time for switching product type H_i to type $H_{i'}$. Terms w_{ij}, p_i, and e_{ij} represent job weight, job processing time, and latest starting time, respectively, to process job h_{ij}.

$$PSA_{h_{ij}h_{i'j'}} = \begin{cases} 0 & \text{if } PSA_{h_{ij}h_{i'j'}} < 0 \text{ or } i = i', j = j', \\ \alpha_1\left(s_{Ui} + s_{i'U} - s_{ii'}\right) + \beta_1\left(\dfrac{w_{ij}}{p_i} + \dfrac{w_{i'j'}}{p_{i'}}\right) + 10Cap\left(\dfrac{\gamma_1}{e_{ij}} - \dfrac{(2-\gamma_1)}{e_{i'j'}}\right) & \text{otherwise.} \end{cases}$$

Step 2. Sort the savings and create a list in descending order of magnitude.

Step 3. Choose the first feasible *K* pairs of jobs on the list satisfying the machine capacity and due date constraints and remove these pairs from the savings list. Then, start the *K* new schedules simultaneously.

Step 4. Starting from the top of the remaining savings list, find the first feasible pair on the list and check which job of the pair is able to be added to one of the two ends of a currently constructed schedule.

Step 5. The chosen jobs form a feasible machine schedule. Repeat Step 4 of Phase I until all the contract jobs are scheduled.

Phase II (spot jobs assigned without extra setup)

Step 1. Calculate the weighted throughput ratio for the spot jobs.

Step 2. Sort the weighted throughput ratio and create a weighted throughput ratio list in descending order of magnitude.

Step 3. Choose the job with highest weighted throughput ratio as the job to be inserted. Find the machine, which has scheduled the jobs with the same product type as the job to be inserted. Whether the job is inserted or not, it

should be removed from the weighted throughput ratio list.

Step 4. If the machine for insertion is found, then insert the chosen job by the side of the job with same product type without violating the machine capacity and due date constraints.

Step 5. Repeat Steps 3 and 4 of Phase II until no more jobs can be found in the weighted throughput ratio list.

Phase III (Remaining spot jobs assigned with extra setup)

Step 1. Calculate the weighted throughput ratio for the remaining spot jobs.

Step 2. Sort the weighted throughput ratio in descending order of magnitude.

Step 3. Choose the top job on the list as the job to be inserted.

Step 4. Schedule the job by applying the cheapest insertion algorithm (Rosenkrantz *et al.* [75]) sequentially to construct a feasible schedule without violating machine capacity and due date restrictions. Whether the job is inserted or not, it should be removed from the weighted throughput ratio list.

Step 5. The algorithm will terminate with no job in the weighted throughput ratio list. Otherwise return to Steps 3 and 4 of Phase III.

To illustrate how the MPSA_TP algorithm may be applied, we consider the PCBSP example with two machines and three product types described in Section 3.2.3. In Phase I, the savings value for each pair between two contract jobs is calculated by Step 1 and shown in Table 3-3 while the values of α_1, β_1, and γ_1, are set to 0.5, 0.05, and 1.5, respectively. The savings are sorted in descending order of magnitude using Step 2, and

the first two feasible pairs (12th pair and 4th pair) are chosen and scheduled in the two machines initially in Step 3. After processing using the algorithm and by repeating Step 4 and Step 5 until no more contract jobs can be found, two schedules are constructed: job h_{C2} precedes job h_{C1} directly on Machine 1 and job h_{B1} precedes job h_{A1} directly on Machine 2.

Table 3-3 The savings values of each contract-jobs pair.

Pair ID	Job pair	Values of savings	Pair ID	Job pair	Values of savings
1	(h_{A1}, h_{B1})	14.75	7	(h_{C1}, h_{A1})	19.14
2	(h_{A1}, h_{C1})	18.29	8	(h_{C1}, h_{B1})	10.89
3	(h_{A1}, h_{C2})	13.57	9	(h_{C1}, h_{C2})	16.21
4	(h_{B1}, h_{A1})	22.76	10	(h_{C2}, h_{A1})	33.28
5	(h_{B1}, h_{C1})	24.54	11	(h_{C2}, h_{B1})	25.02
6	(h_{B1}, h_{C2})	19.83	12	(h_{C2}, h_{C1})	35.06

In Phase II, the three spot jobs (h_{A2}, h_{A3}, and h_{B2}) are inserted into the two constructed schedules without violating machine capacity and due date constraints. In Step 1 of this phase, the weighted throughput ratios of the three jobs are computed: 1.9, 1.9, and 2 for job h_{A2}, job h_{A3}, and job h_{B2}, respectively. The weighted throughput ratios are sorted using Step 2. It was found that h_{B2} is the highest ratio and is, therefore, chosen and removed from the weighted throughput ratio list in Step 3. Machine 2 has the same product type as job h_{B2}; therefore, job h_{B2} can be inserted into Machine 2 in Step 4. However, the insertion of job h_{B2} into the two possible positions (preceding or following job h_{B1} immediately) will cause the job h_{A1} to violate its due date. Hence, job h_{B2} is not inserted into this machine. Then, Step 3 of the algorithm is repeated and the other spot job h_{A2} is considered. In Step 4, Machine 2 is the inserted machine. Job h_{A2} is scheduled following job h_{A1} without any extra setup time and without violating any restrictions. Furthermore, by repeating Step 3, the third spot job h_{A3} is chosen but is not

suitable for insertion due to its violation of due date and machine capacity.

Finally, in Phase III, the remaining spot jobs (h_{B2} and h_{A3}) are considered and scheduled although extra setup times are required. The weighted throughput ratios of the two spot jobs are computed and sorted in descending order in Step 1 and Step 2 of Phase III, respectively. In Step 3, job h_{B2} is chosen and removed from the list, but it cannot be inserted into the two machines. However, after processing is repeated in Step 3, the other job, h_{A3}, is inserted into Machine 1 using the cheapest insertion algorithm in Step 4. Due to the absence of more jobs on the weighted throughput ratio list and fullness of the two machines, the final solution from the PSA_TP algorithm is therefore 316 and is depicted in Figure 3-5.

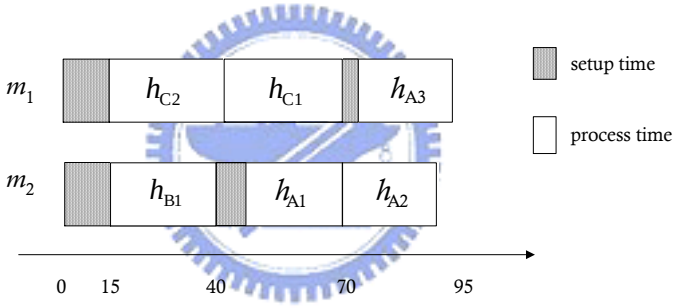


Figure 3-5 Gantt chart for the example problem.

3.4.3.2. Three-phase Modified Generalized Savings Algorithm

The MGSA_TP constructs the schedules sequentially in contrast to the MPSA_TP which creates a multiple of K machine schedules simultaneously. In addition, the difference in Phase I with MPSA_TP is that MPSA_TP considers only two end points when merging a new job into the current partial schedule, while MGSA_TP considers not only the end points but also the positions between two consecutive jobs when merging a new job into the current partial schedule. The savings function is expressed as follows:

$$GSA_{h_{ij}h_{i'j'}} = \begin{cases} 0 & \text{if } GSA_{h_{ij}h_{i'j'}} < 0 \text{ or } i = i', j = j', \\ \alpha_2\left(s_{Ui} + s_{i'U} - s_{ii'}\right) + \beta_2\left(\dfrac{w_{ij}}{p_i} + \dfrac{w_{i'j'}}{p_{i'}}\right) + 10Cap\left(\dfrac{\gamma_2}{e_{ij}} - \dfrac{(2-\gamma_2)}{e_{i'j'}}\right) & \text{otherwise.} \end{cases}$$

Based on the new savings function, the initial partial schedule can be selected from the top of the savings list. In addition, schedules can be expanded based on parameters δ_1, δ_2, and formulas (3-21)-(3-24) (designed by Christofides *et al.* [23]). Let *PS* be the current schedule, $PS=\left(u_U,...,u_{g-1},u_g,...,u_G,u_{U'}\right)$, where u_U and $u_{U'}$ are virtual jobs (machine idle status). The insertion costs are computed using formulas (3-21)-(3-22) for every unscheduled job h_{ij} at every possible position of *PS*. Let $\lambda\left(u_{g-1},h_{ij},u_g\right)$ be the additional set-up cost when job h_{ij} is inserted between position (*g-1*) and *g* in schedule *PS*. Let $\lambda^*\left(u_{g-1},h_{ij},u_g\right)$ be the minimal insertion cost value.

$$\lambda\left(u_{g-1},h_{ij},u_g\right) = s_{u_{g-1}i} + s_{iu_g} - \delta_1 \times s_{u_{g-1}u_g}, \qquad 1 \le \delta_1 \le 2, \tag{3-21}$$

$$\lambda^*\left(u_{g-1},h_{ij},u_g\right) = \min_{g=1,...,G}\left[\lambda\left(u_{g-1},h_{ij},u_g\right)\right]. \tag{3-22}$$

Job h_{ij} is chosen, which maximizes the savings $\sigma^*\left(u_{g-1},h_{ij},u_g\right)$ while minimizing the insertion cost $\lambda^*\left(u_{g-1},h_{ij},u_g\right)$, and which avoids the algorithm to create a new schedule on another machine with a high setup time $\delta_2 \times s_{Ui}$. Furthermore, in addition to taking into account machine capacity, the due date constraints of all jobs must also be examined for violation before a job is inserted. The procedure is repeated until all schedules are full and cannot be expanded. Phase II and Phase III are same as the corresponding phases of the MPSA_TP algorithm.

$$\sigma\left(u_{g-1},h_{ij},u_g\right) = \delta_2 \times s_{Ui} - \lambda^*\left(u_{g-1},h_{ij},u_g\right), \qquad 1 \le \delta_2 \le 2, \tag{3-23}$$

$$\sigma^*\left(u_{g-1},h_{ij},u_g\right) = \max\left[\sigma\left(u_{g-1},h_{ij},u_g\right)\right]. \tag{3-24}$$

The same illustrative example described in Section 3.2.3 with two machines and three product types is also used to illustrate Phase I of the MGSA_TP. First, the savings are computed when the values of α_1, β_1, and γ_1, are set to 0.5, 0.05, and 1.5, respectively. The savings values for the MGSA_TP are the same as those obtained by the MPSA_TP, as shown in Table 3-3. In contrast to the MPSA_TP algorithm, one job pair (h_{C2}, h_{C1}) with the largest savings is chosen and scheduled on Machine 1 initially. Second, the other two contract jobs (h_{A1} and h_{B1}) should be considered with their insertion and savings cost using formulas (3-21)-(3-24) in order to decide which one should be inserted into the current partial schedule. For the insertion costs, there are three possible positions for each candidate contract job on the current partial schedule, PS=$(u_U, h_{C2}, h_{C1}, u_{U'})$. Therefore, six insertion costs are computed while $\delta_1 = 2$ and $\delta_2 = 1$ and are presented in Table 3-4. In Table 3-4, each minimal insertion cost is obtained for h_{A1} and h_{B1} are -8 and -10, respectively. The following savings costs are then obtained, $\sigma(u_{g-1}, h_{A1}, u_g) = 1\times 15 - (-8) = 23$ and $\sigma(u_{g-1}, h_{B1}, u_g) = 1\times 15 - (-10) = 25$ to h_{A1} and h_{B1}, respectively. Therefore, the $\sigma^*(u_{g-1}, h_{ij}, u_g) = \max[23, 25] = 25$, then the job h_{B1} is chosen.

Table 3-4 The insertion cost of each job at every possible position.

Job ID	Possible insertion positions	Insertion cost	Values
h_{A1}	$\lambda(u_U, h_{A1}, h_{C2})$	$s_{UA} + s_{AC} - 2\times s_{UC}$	-8
h_{A1}	$\lambda(h_{C2}, h_{A1}, h_{C1})$	$s_{CA} + s_{AC} - 2\times s_{CC}$	10
h_{A1}	$\lambda(h_{C1}, h_{A1}, u_{U'})$	$s_{CA} + s_{AU'} - 2\times s_{CU'}$	3
h_{B1}	$\lambda(u_U, h_{B1}, h_{C2})$	$s_{UB} + s_{BC} - 2\times s_{UC}$	-10
h_{B1}	$\lambda(h_{C2}, h_{B1}, h_{C1})$	$s_{CB} + s_{BC} - 2\times s_{CC}$	21
h_{B1}	$\lambda(h_{C1}, h_{B1}, u_{U'})$	$s_{CB} + s_{BU'} - 2\times s_{CU'}$	16

Third, job h_{B1} is the candidate job and will be inserted on Machine 1 preceding h_{C2}. However, this insertion would cause h_{C2} and h_{C1} to be out of their due dates. Therefore, job h_{B1} is not inserted in the machine. As in this example there is only one

other possible contract job (h_{A1}), the process needs to be repeated to determine if it is suitable for insertion. However, it was found that job h_{A1} also cannot be inserted in Machine 1 because of due date and machine capacity constraints. Therefore, the algorithm steps are repeated on Machine 2, jobs h_{B1} and h_{A1} are scheduled on Machine 2 as they were found not to violate any restrictions. Phase I of MGSA_TP is terminated when all contract jobs are scheduled.

3.5. Test Problems Design

For the purpose of testing and comparing the performance of the proposed two new algorithms on various PCBSP with different data characteristics, we generate a set of twenty-four problems, which are taken from a module assembly factory located on the HsinChu Science-Based Industrial Park in Taiwan. For the test problems investigated, jobs of twenty-six product types contain contract and spot jobs. The jobs are scheduled to five identical parallel machines. The contract jobs must be completed on the parallel machines before the given due dates. The machine capacity is set to 4320 minutes, which is set to equal to the planning period (three days). 'Minute' here is used as the time unit for the job process time, setup time, due date, and machine capacity.

The structure and data of the generated test problems are generated covering most real-world applications. These characteristics include: (1) workload level of contract jobs, (2) tightness of due dates, (3) setup time variation, and (4) variation of (contract/spot) weight ratio. These problem sizes range from low workload level of contract jobs, loose due date tightness, small setup time variation, low variation of (contract/spot) weight ratio, to high workload level of contract jobs, tight due date tightness, high setup time variation, and high variation of (contract/spot) weight ratio.

3.5.1. Workload Level of Contract Jobs

In the real production environment, different workload levels of contract jobs result from different market demand or sale seasons, such as hot seasons in electronic industries. Therefore, we need to evaluate the impact of different workload levels of contract jobs on the performance of the solution algorithms. Owing to varied workload levels of contract jobs, the number of contract jobs is different. Let ES be the estimated setup time required in the problem. $AVG_i[s_{ii'}]$ is the average setup time from product type H_i to other types. And finally, $AVG[s_{iU}]$ is the average setup time to switch to idle status, $AVG[s_{Ui}]$ be the average setup time from idle status to process. The estimated setup time can be expressed as follows.

$$ES = K(AVG[s_{Ui}] + AVG[s_{iU}]) + \frac{(I-1)}{I}\sum_{all\ i} AVG_i[s_{ii'}]. \qquad (3\text{-}25)$$

The workload calculation formula in our investigation can be expressed in Equation (3-26). In the twenty-four testing problems, each problem contains 120 jobs carrying specific contract jobs and spot jobs. Taking problem 4, 5, and 6 (see Table 3-5) for example, when the number of contract jobs is 25, 50, and 75, the workload level of contract jobs will be 42%, 64%, and 86%, respectively. Problem 6 is used to illustrate how the calculation of workload level be applied, a setup time matrix is presented in Table A1 and detailed job information is shown in Table A2 (see Appendix). The average setup times required for switching from a job with idle status to process ($AVG[s_{Ui}]$) is 120 minutes, while the reverse ($AVG[s_{iU}]$) only requires 0 minutes. The average setup time ($AVG[s_{ii'}]$) requires for switching from all contract jobs of product type H_i to type $H_{i'}$ is equal to 3496.9 minutes. Therefore, the estimated setup time is 3962.4 minutes. Furthermore, the total processing time of the contract jobs in Problem 6 is 14451 minutes.

The workload level of Problem 6 is then obtained using Equation (3-26) when the number of machines (K) is 5 and each machine capacity (Cap) is 4320 minutes.

$$Workload\ level = \frac{ES + \sum_{i}^{I} \sum_{j}^{J_i^{contract}} p_i}{K \times Cap} \times 100\% \quad \text{for } i = 1,2,\ldots,I \text{ and } j = 1,2,\ldots,J_i^{contract}. \tag{3-26}$$

3.5.2. Tightness of Due Dates

To analyze the impact of the tightness of due dates on the performance of scheduling algorithms, we include two levels of the tightness of due dates. Here, we apply the tightness index formula proposed by Pearn *et al.* [68] and Equation (3-25) to estimate the setup time. In the tight situation, the number of jobs during the due dates of day1 and day2 are greater than day 3. In the loose situation, the number of jobs during the due date of day 1 would be less than the number of jobs during the due dates of day 2 and day3.

Table 3-5 Summarized information of 24 problems.

Problem Number	Tightness of due date		Workload level of contract jobs			Setup time variation		Variation (contract /spot) weight ratio	
parameter	Tight	Loose	Low	Middle	High	Small	Large	Small	Large
1	•		•				•		•
2	•			•			•		•
3	•				•		•		•
4	•		•				•	•	
5	•			•			•	•	
6	•				•		•	•	
7		•	•				•		•
8		•		•			•		•
9		•			•		•		•
10		•	•				•	•	
11		•		•			•	•	
12		•			•		•	•	
13	•		•			•			•
14	•			•		•			•
15	•				•	•			•
16	•		•			•		•	
17	•			•		•		•	
18	•				•	•		•	
19		•	•			•			•
20		•		•		•			•
21		•			•	•			•
22		•	•			•		•	
23		•		•		•		•	
24		•			•	•		•	

3.5.3. Setup Time Variation

In PCBSP, we reduce setup time by scheduling contract and spot jobs without violating the contracted due dates. Thus, the setup time is one of the critical factors for increasing the impact of results. However, the setup time could be varied because of different considerations of setup operations. For instance, the cool down and the rapid rising of temperature and voltage are good examples of differing conditions. In our test, we include two levels of setup time variation. The high setup time variation is 10519.4 and the low setup time variation is 3990.3.

3.5.4. Variation of (Contract/Spot) Weight Ratio

The contract/spot weight ratio is the division of the contract job weight by the corresponding spot job weight for each of the product types. The variation of the contract/spot weight ratio is to analyze the variance among different product types. In the real world application, the ratio among different product types should be varied owing to the market competition. The high variation of contract/spot weight ratio is set to 0.01 and the low variation of contract/spot weight ratio is set to 0.001.

The problem sets of the four considered factors have 24 different problems. The problem lists with the four different factors are listed as Table 3-5. Take problem 6 for example, the setup time matrix is presented in Table A1 and the detailed job information is shown in Table A2 (see Appendix).

3.6. Computational Results

To solve the PCBSP, two heuristic algorithms are coded in Virtual Basic 6.0 programming language, and run on a Pentium IV 3.2GH$_Z$ PC. We first experiment with the two new algorithms on ten small size of the PCBSP, where the optimal solutions are

available. The size of the problems range from 10 to 15 total jobs, 6 to 10 contract jobs, and 950 to 1650 minutes machine capacity with various workload levels of contract jobs. For these problems, we write a C++ programming code to generate the constraints and variables of the models. In addition, we solve them using the IP software CPLEX 7.5 to obtain optimal solutions. The computational results are displayed in Table 3-6.

Table 3-6 indicates that the heuristic solutions receive 8 optimal solutions (out of 10) in term of total weight throughput. The average gap between the optimality and two heuristic algorithms, MGSA_TP and MPSA_TP, are 1.4% and 0.5%, respectively. The average run times (in CPU seconds) for the two heuristic algorithms are indeed significantly faster than the run time of optimality.

Table 3-6 A comparison between the optimal solutions and two heuristic algorithms (underlines indicate the optimal solutions).

Prob. No.	J	$J^{contract}$	Cap	Optimal value	CPU (sec)	MGSA_TP		MPSA_TP	
						weighted throughput	CPU (sec)	weighted throughput	CPU (sec)
1	10	6	1000	444000	501.64	444000	0.031	444000	0.016
2	11	6	950	444000	405.09	444000	0.031	444000	0.016
3	12	5	1100	514000	451.89	514000	0.031	514000	0.016
4	12	8	1075	557000	129.75	557000	0.031	557000	0.016
5	13	6	1000	488000	574.39	452000	0.047	471000	0.031
6	13	6	1100	537000	224.61	537000	0.031	537000	0.016
7	14	8	1300	660000	5920.83	660000	0.047	660000	0.031
8	14	7	1200	597000	1219.03	555000	0.047	584000	0.031
9	15	9	1440	670000	13430.4	670000	0.048	670000	0.032
10	15	10	1650	760000	61640.1	760000	0.063	760000	0.031

For large size of the PCBSP, solving the optimal solutions using integer programming model is computationally inefficiently. Therefore, in the following, we test the performance of the two new algorithms on the twenty-four problems described in Section 3.5. As the two parameters of the two algorithms, α_1 and γ_1, have been applied by Pearn *et al.* [68], the preliminary comparison based on the two parameters analysis of MPSA_TP is summaries in Table 3-7. We first examine the three values of α_1 and five values of γ_1, which are initially set to α_1 = 0, 0.5, 1 and γ_1 =0, 0.5, 1, 1.5, 2,

respectively. Therefore, fifteen different combinations are obtained by various α_1 and γ_1 values where β_1 is set to 0. Table 3-7 displays that the average values of the solutions obtained on 24 problems described previously. It appeared that the best parameter combination (α_1, γ_1) is (0.5, 1.5) in terms of average weighted throughput. Therefore, we limit the two parameters and further analysis the settings of parameter β_1 that is a new-added parameter for PCBSP. In this dissertation, parameter β_1 is examined to check whether parameter β_1 affects the solution for PCBSP ($0 < \beta_1 \le 1$) or not ($\beta_1 = 0$). We set $\beta_1 = 0, 0.05, 0.1, \ldots, 0.95,$ and 1 and display the computational results in Table 3-8. Table 3-8 indicates that the parameter combination, (α_1, β_1, γ_1) = (0.5, 0.05, 1.5), outperforms than the other settings. We therefore set the choice of parameter values of the MPSA_TP to (α_1, β_1, γ_1) = (0.5, 0.05, 1.5).

Table 3-7 The preliminary comparison with various parameter settings.

α_1	β_1	γ_1	Average weighted throughput	α_1	β_1	γ_1	Average weighted throughput	α_1	β_1	γ_1	Average weighted throughput
0	0	0	5284083	0.5	0	0	5568292	1	0	0	5627813
0	0	0.5	5326438	0.5	0	0.5	5656500	1	0	0.5	5670188
0	0	1	5341271	0.5	0	1	5657396	1	0	1	5670521
0	0	1.5	5341271	0.5	0	1.5	5699958	1	0	1.5	5688750
0	0	2	5336375	0.5	0	2	5674292	1	0	2	5677563

We obtained 72 computational results, which include 48 results for MPSA_TP with two types of parameter combination (MPSA_TP denotes MPSA_TP with $\alpha_1 = 0.5$, $\beta_1 = 0.05$, and $\gamma_1 = 1.5$; MPSA_TP2 denotes MPSA_TP with $\alpha_1 = 0.5$, $\beta_1 = 0$, and $\gamma_1 = 1.5$). MPSA_TP2 represents that the weighted throughput ratio adding item of the savings calculation is not considered. Table 3-9 displays the detailed comparison within the two types of algorithms. It denotes the number of better solutions comparing to the two proposed heuristic procedures. In comparing the three algorithms, the test results showed that:

Table 3-8 The comparisons with different parameter values of β_1 when $\alpha_1 = 0.5$ and $\gamma_1 = 1.5$ (underlines indicate the best solutions for each problem instance).

Prob. No.	β_1																					
	0	0.05	0.1	0.15	0.2	0.25	0.3	0.35	0.4	0.45	0.5	0.55	0.6	0.65	0.7	0.75	0.8	0.85	0.9	0.95	1
1	493050	515050	505300	506050	506900	503900	502650	503050	499300	496500	497050	505300	505300	505300	505300	494300	503200	503200	503200	507300	507300
2	535100	547000	544400	543750	537400	544150	540350	538500	549200	548200	541500	540850	537350	533150	535500	535500	538500	538500	538450	527750	533950
3	603850	600950	584250	588600	582750	577300	573950	587000	576250	583750	572650	581050	577850	580750	585150	585150	581300	578050	578050	578050	578050
4	561450	591050	577500	579250	579100	576100	574850	580250	570500	566700	580450	578300	578300	578300	578300	565300	575400	575400	575400	581300	581300
5	577600	591400	589600	589700	597800	597900	587600	596600	585700	588700	588700	595600	592700	586050	586050	599150	580850	579900	579900	582050	582050
6	629850	625950	607250	609600	601750	600300	594950	610000	597250	604750	591650	603050	601850	601750	607150	607150	604300	600050	600050	600050	600050
7	501300	519300	523900	507800	507900	507900	511900	498750	499300	499300	507300	511200	499100	503200	503200	503200	495750	495750	495750	495750	495750
8	552700	558400	552700	545850	553000	557700	551100	545050	556400	557200	560400	557400	552150	545050	556050	553700	553700	553700	546300	545400	542200
9	574300	599700	595400	586150	589650	587800	583400	594100	581350	601950	578600	578600	580350	570450	575450	575450	584300	584300	584300	584300	575050
10	581500	590500	601100	581100	581100	581100	591100	570050	570500	570500	585500	585400	575400	575400	575400	575400	567750	567750	567750	567750	567750
11	599900	620900	580700	601150	599150	618000	613000	610000	604000	608000	593800	584600	587700	587150	587150	586550	589050	589050	589050	602900	577300
12	593300	622700	618400	608150	611650	609800	603400	616100	601350	625950	598600	598600	600350	589450	596450	596450	605300	605300	605300	605300	595050
13	497300	507350	513050	498350	506900	507900	511550	503050	512500	507300	505300	507100	507100	507050	507050	507100	511200	511200	511200	494300	494300
14	542800	556300	548400	548400	553000	549400	542600	551000	551000	547600	551200	551200	545700	544400	536050	536050	536950	536950	531450	536400	522750
15	592750	602850	587250	600050	582450	590700	588450	586750	585850	589850	587200	588650	584950	586100	586100	586100	581850	581650	581650	584550	583350
16	571700	585550	582250	574750	568800	581100	585750	580250	586700	580500	582500	580400	580400	585150	585150	580400	585400	585400	585400	565300	565300
17	586600	614700	610000	605900	596000	599800	595800	595800	601000	592950	595600	591400	605800	592700	592700	592050	587050	580850	580850	576850	576850
18	614750	627350	608250	625050	604450	614700	614450	606750	608850	615850	610200	613650	609250	607300	607300	607300	601850	604650	604650	608550	607350
19	506300	519200	510900	515900	511900	511900	506900	510950	507200	503200	507300	507000	507050	507050	511300	503200	503200	503200	503200	503200	503200
20	553700	564000	547050	563000	554100	557850	554200	553000	560400	554050	554050	552600	546950	550550	550550	549150	549150	549150	545200	546200	546200
21	601350	602700	594450	589450	595250	593650	594650	597950	601200	596850	596850	593050	587100	587100	582450	589050	583250	583250	583250	583250	583250
22	583500	595200	584100	591100	581100	581100	581100	580150	585500	581300	580500	585200	589050	589050	590800	579500	579500	579500	579500	579500	579500
23	599900	605200	597800	602700	605000	605800	606500	606900	598900	596050	602900	590050	586950	586950	591150	591150	597800	596150	597150	586050	586050
24	625350	626700	617450	613450	618250	615650	616650	620950	626200	620850	620850	615050	609100	609100	603450	611050	604250	604250	604250	604250	604250
*	2	12	3	0	0	0	0	1	2	2	1	0	0	0	0	1	0	0	0	0	0

The line labeled by symbol of * represents the number of best solutions for each problem instance selected from all of the parameter combinations.

Table 3-9 Performance comparisons in the three algorithms.

Parameter	Weighted throughput		
	MGSA_TP	MPSA_TP	MPSA_TP2
Prob No.	$\alpha_2 = 0.3$ $\beta_2 = 0.02$ $\gamma_2 = 0.4$ $\delta_1 = 2$ $\delta_2 = 1$	$\alpha_1 = 0.5$ $\beta_1 = 0.05$ $\gamma_1 = 1.5$	$\alpha_1 = 0.5$ $\beta_1 = 0$ $\gamma_1 = 1.5$
1	5056000	**5150500**	4930500
2	5385000	**5470000**	5351000
3	5796500	6009500	**6038500**
4	5776000	**5910500**	5614500
5	5791000	**5914000**	5776000
6	5986500	6259500	**6298500**
7	5099000	**5193000**	5013000
8	5440000	**5584000**	5527000
9	**6020000**	5997000	5743000
10	5826000	**5905000**	5815000
11	5915000	**6209000**	5999000
12	**6236000**	6227000	5933000
13	**5102000**	5073500	4973000
14	5499000	**5563000**	5428000
15	**6037000**	6028500	5927500
16	5786000	**5855500**	5717000
17	5959000	**6147000**	5866000
18	**6277000**	6273500	6147500
19	5016000	**5192000**	5063000
20	5577000	**5640000**	5537000
21	6006000	**6027000**	6013500
22	5726000	**5952000**	5835000
23	6014000	**6052000**	5999000
24	6224000	**6267000**	6253500

Result with bold is the best solution among the three algorithms.

(1) The MPSA_TP received 19 better solutions (out of 24) than the MGSA_TP. The average improvement between the 19 improved problems of the MPSA_TP comparing with the MGSA_TP in terms of weighted throughput is 2.2%.

(2) In the MPSA_TP algorithm group, the MPSA_TP received 22 better solutions (out of 24) than the MPSA_TP2. The weighted throughput ratio term, in the savings calculation of the MPSA_TP, indeed improves the solutions.

By computing the mean of the solutions generated by each algorithm in total weighted throughput for the twenty-four problems, we could compare the performances among these algorithms. To further analyze the performance of those algorithms on problems with different characteristics, we grouped the results with the four problem factors and factor levels, which is shown in Table 3-10. Since the factors such as tightness of due date, setup time variation, and variation of contract/spot weight ratio, contain two levels of values, these groups each include 12 results. Because the factor of workload level of contract jobs contains three levels of values, these groups include 8 results each.

In table 3-10, the MPSA_TP outperforms the MGSA_TP and the MPSA_TP2 on all nine groups. Therefore, we can say the performance of the MPSA_TP is better than the MGSA_TP and the MPSA_TP2 stably in varied situations.

Table 3-10 Results in means with different problem characteristic groups.

Algorithm		MGSA_TP	MPSA_TP	MPSA_TP2
Parameter		$\alpha_2 = 0.3,\ \beta_2 = 0.02,\ \gamma_2 = 0.4$ $\delta_1 = 2,\ \delta_2 = 1$	$\alpha_1 = 0.5,\ \beta_1 = 0.05,\ \gamma_1 = 1.5$	$\alpha_1 = 0.5,\ \beta_1 = 0,\ \gamma_1 = 1.5$
	n	Mean	Mean	Mean
Total	24	5731250	5829167*	5699958
Tightness DD= Tight	12	5704250	5804583*	5672333
Tightness DD= Loose	12	5758250	5853750*	5727583
Contracted workload= Low	8	5423375	5529000*	5370125
Contracted workload= Middle	8	5697500	5822375*	5685375
Contracted workload= High	8	6072875	6136125*	6044375
Setup time variation = Small	12	5768583	5839250*	5730000
Setup time variation = Large	12	5693917	5819083*	5669917
Variation of weight ratio= Small	12	5502792	5577333*	5462083
Variation of weight ratio= Large	12	5959708	6081000*	5937833

Result with * is best solution among the three algorithms.

Finally, we compared performances generated by the three algorithms, which is presented in Table 3-11, with respect to (1) average rank among the three algorithms, (2) number of problems receiving the best solutions, and (3) average run times in CPU seconds on a Pentium IV 3.2GH_Z PC.

Table 3-11 Performance comparisons of the three algorithms (24 problems).

	MGSA_TP	MPSA_TP	MPSA_TP2
Average rank among the three algorithms	2.125	1.25	2.625
Number of problems receiving the best solutions	5	17	2
Average run times CPU seconds	5.11	11.9	8.08

The results, displayed in Table 3-11, indicate that the run times of the three algorithms are quite fast for solving those problems containing five machines and 120 jobs with different problem characteristics. The results also reveal that the MPSA_TP outperformed the other algorithms in terms of average rank among the three algorithms and number of problems receiving the best solutions. It may be explained by that the MPSA_TP creates a multiple of K machine schedules simultaneously at the first phase which may cause that job pairs with the same product type are assigned to the same machines easily. Therefore, the setup times incurred from different product types can be reduced and then the values of weighted throughput will be increased. However, in Table 3-11, the run times of MPSA_TP are larger than the other two algorithms. The most likely explanation for this is that the number of job candidates which need to be checked their probable insertions and be determined the appropriate positions for insertions is more than the others.

.

4. Solutions for the Aging Test Scheduling Problem

In this chapter, the aging test scheduling problem (ATSP) with a minimum makespan criterion, is presented and solved. First, a mixed integer linear programming (MILP) model for the scheduling problem is provided. Secondly, a compound MILP-based algorithm is proposed to determine the number of batches and to apply this number as one parameter in the MILP model in order to reduce the complexity of the problem. Then, three heuristic algorithms are given to solve the large-scale aging test scheduling problem and computational comparisons are offered.

4.1. Introduction

The existing and growing importance of parallel batch processing machines demands a solution to its scheduling problem in order to improve efficiency of production. In this chapter, the aging test scheduling problem (ATSP) with a minimum makespan criterion is presented, which is a parallel batch processing machine scheduling problem. At the end of the module assembly process (see Figure 4-1), the aging test is undertaken by the only batch server in the whole process to put the assembled modules with different product families and unequal ready times into high temperature parallel batch processing machines. This batch production type is referred to as compatible product families. The batch processing times and batch ready times in the aging test operation are dependent on the longest processing time and the latest ready time of all the jobs in each batch, respectively. Since the jobs with unequal ready times and long batch dependent processing times are processed at the end of the module assembly, it is

essential that the development of scheduling algorithms for the aging test scheduling problem with a minimum makespan criterion.

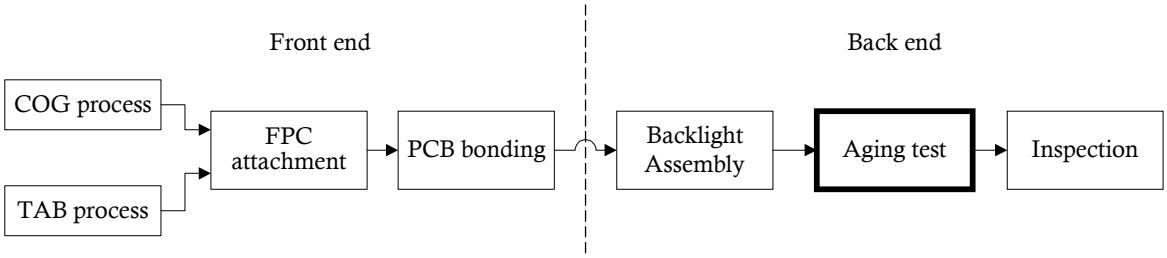


Figure 4-1 The six steps in the module assembly process.

The ATSP is a multi-dimensional parallel batch processing machine scheduling problem; the relationships among these dimensions is depicted as Figure 4-2. The constraints of unequal ready times and non-identical job sizes affect the determination of the number of batches which is based on the limited machine capacity and which then determines the batch processing times. Whenever batch formations are altered, the batch processing times are consequently varied and the batch sequence needs to be rescheduled in order to minimize the makespan and to delivery the jobs to customers. The ATSP involves the constraints of unequal ready times, non-identical job sizes, limited machine capacity, and batch dependent processing times which is a variation of the classical parallel batch processing machine scheduling problem considered by Lee and Uzsoy [51] and Chang *et al.* [14]. Since the aging test scheduling problem involves batch formation and scheduling with real-world constraints, it is an intractable problem for industrial planners and theoretical researchers. Therefore, the development of efficient algorithms to form appropriate batches and to arrange a suitable schedule for the ATSP is difficult but critical.

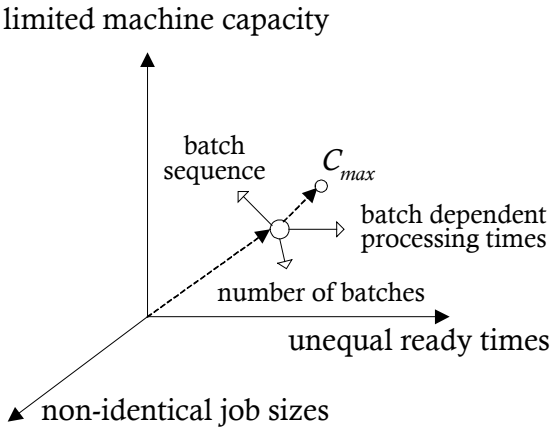


Figure 4-2 The multiple dimensions of the aging test scheduling problem.

To our best knowledge, the aging test scheduling problem with unequal ready times and non-identical job sizes has not been considered by the other researchers in this area. This problem can be represented by $P/\text{batch}, r_j, \text{compatible}/C_{\max}$, and involves batch formation and scheduling simultaneously. In this dissertation, it is assumed that the ready times of jobs are known before the determination of the parallel batch processing machines schedule. The scheduling problem is formulated as a mixed integer linear programming (MILP) model to minimize the makespan. The programming model considers machine capacity restrictions, unequal ready times, non-identical job sizes, and batch-dependent processing times, in order to reflect real situations more accurately. A compound MILP-based algorithm is developed to determine the number of batches and to apply this number as one parameter in running the proposed MILP model. Furthermore, three efficient heuristic algorithms are also proposed. The best solution selected from the result of the MILP model and that of the compound algorithm is used here as a convenient reference point to assess the accuracy of the heuristic solutions. To demonstrate the effectiveness and efficiency of all the proposed algorithms, a set of testing problems is explored and a real-world problem is taken from a module assembly

shop floor in a TFT-LCD factory located in the Science-based Industrial Park in Hsinchu, Taiwan.

4.2. An Integer Programming Formulation

The mixed integer linear programming (MILP) model for the ATSP with unequal ready times and non-identical job sizes in order to minimize the makespan is formulated in this section and is referred to as Model P. A set of jobs is given to be processed in batches by identical parallel machines. Let the total number of jobs be denoted by N and the number of batches be denoted by B. In this dissertation, however, an individual job cannot be split into different batches due to the inconvenience to practical management that might result. Let machine group $M = \{m_k \mid k = 1, 2, ..., K\}$, contain the K parallel batch processing machines. Due to the fact that the unequal ready times and batch dependent processing times are considered, they are associated with job j and have a processing time denoted by p_j and a ready time denoted by r_j. The batch processing time may vary, depending on the composite jobs. Term pt_b is the longest processing time of all the jobs processed simultaneously in the b^{th} batch, and it represents the batch processing time. The batch ready time is also the latest ready times of those composite jobs. Each job has a non-identical job size (s_j). A batch can be processed on a machine on the condition that the accumulated size of those jobs in that batch does not exceed the machine's capacity (a maximum number of pieces can be processed simultaneously on a machine) (S'). Each job in its associated batch is a candidate that is processed without preemption on one machine. The aging test scheduling problem is to form batches appropriately as well as to find a schedule for those batches that satisfies the ready time restrictions without violating the machine capacity constraints, while also achieving the objective of minimizing makespan. Initially, it is assumed that each job will be contained

in an individual batch ($B = N$). However, after the mathematical model is solved, it may be that the better solution might require combining more than one job into one batch. Hence, by using the proposed MILP model, the number of batches required will be self-evident. Before the MILP model (Model P) is presented, the notations used in the formulation are listed below.

Indices:

j: job index, $j = 1,2,\ldots, N$,

b: batch index, $b = 1,2,\ldots, B$,

k: machine index, $k = 1,2,\ldots, K$.

Decision variables:

x_{jbk}: $\begin{cases} 1 & \text{if job } j \text{ is assigned to batch } b \text{ on machine } m_k, \\ 0 & \text{otherwise;} \end{cases}$

$y_{bb'k}$: $\begin{cases} 1 & \text{if batch } b' \text{ is scheduled following batch } b \text{ on machine } m_k, \\ 0 & \text{otherwise;} \end{cases}$

z_{bk}: $\begin{cases} 1 & \text{if batch } b \text{ is assigned to machine } m_k, \\ 0 & \text{otherwise;} \end{cases}$

t_{bk} : the starting time of batch b to be processed on machine m_k ,

$C_{\max}$: the maximum completion time (makespan).

Model P:

$$\text{Minimize} \quad C_{\max} + \mu \sum_{b=1}^{B} \sum_{k=1}^{K} z_{bk} \tag{4-1}$$

subject to

$$\sum_{b=1}^{B} \sum_{k=1}^{K} x_{jbk} = 1, \quad \text{for all } j, \tag{4-2}$$

$$\sum_{k=1}^{K} z_{bk} \le 1, \quad \text{for all } b, \tag{4-3}$$

$$\sum_{j=1}^{N} x_{jbk} \le Q_1 z_{bk}, \quad \text{for all } b, k, \tag{4-4}$$

$$\sum_{j=1}^{N}\sum_{k=1}^{K} s_j x_{jbk} \le S', \quad \text{for all } b, \tag{4-5}$$

$$pt_b \ge p_j \times x_{jbk}, \quad \text{for all } j, b, k, \tag{4-6}$$

$$C_{\max} \ge t_{bk} + pt_b, \quad \text{for all } b, k, \tag{4-7}$$

$$t_{bk} \ge r_j x_{jbk}, \quad \text{for all } j, b, k, \tag{4-8}$$

$$t_{bk} + pt_b - t_{b'k} + Q_2(y_{bb'k} - 1) \le 0, \quad \text{for all } b, k, b' \ne b, \tag{4-9}$$

$$(y_{bb'k} + y_{b'bk}) - Q_2(z_{bk} + z_{b'k} - 2) \ge 1, \quad \text{for all } b, k, b' \ne b, \tag{4-10}$$

$$(y_{bb'k} + y_{b'bk}) + Q_2(z_{bk} + z_{b'k} - 2) \le 1, \quad \text{for all } b, k, b' \ne b, \tag{4-11}$$

$$(y_{bb'k} + y_{b'bk}) - Q_2(z_{bk} + z_{b'k}) \le 0, \quad \text{for all } b, k, b' \ne b, \tag{4-12}$$

$$(y_{bb'k} + y_{b'bk}) - Q_2(z_{b'k} - z_{bk} + 1) \le 0, \quad \text{for all } b, k, b' \ne b, \tag{4-13}$$

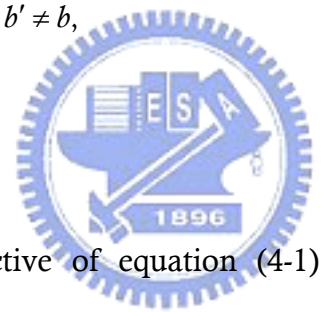
$$(y_{bb'k} + y_{b'bk}) - Q_2(z_{bk} - z_{b'k} + 1) \le 0, \quad \text{for all } b, k, b' \ne b, \tag{4-14}$$

$$x_{jbk} \in \{0, 1\}, \quad \text{for all } j, b, k, \tag{4-15}$$

$$y_{bb'k} \in \{0, 1\}, \quad \text{for all } b, k, b' \ne b, \tag{4-16}$$

$$z_{bk} \in \{0, 1\}, \quad \text{for all } b, k, \tag{4-17}$$

$$C_{\max} \ge 0. \tag{4-18}$$



The bi-functional objective of equation (4-1) of Model P is to minimize the maximum completion time and the number of batches. The former is the main objective for the scheduling problem and the latter is the subsidiary one in order to reduce the complexity of the batch sequence. Therefore, the term μ is a constant, which is chosen to be a sufficiently small value which cannot affect the makespan. Constraint (4-2) guarantees that each job is assigned to one batch and processed on exactly one machine. Constraint (4-3) ensures that each batch is either processed once by one machine or not at all. Constraint (4-4) is a contingent constraint. That is, if some jobs are assigned to batch b on machine m_k (x_{jbk}=1), then batch b should be assigned to machine m_k (z_{bk}=1). Term Q_1 is a constant and is greater than the total number of jobs (N). Constraint (4-5) is the batch size constraint, which requires that the sum of all the pieces of each job

contained in each batch on one machine be simultaneously processed and within the maximum machine capacity. Constraint (4-6) ensures that the processing time of each batch is the longest processing time of all the jobs simultaneously processed in a batch. Constraint (4-7) is the maximum completion time (makespan) and is always greater than or equal to the sum of the starting and processing times for each batch. Constraint (4-8) indicates that the starting time of each batch is greater than or equal to the ready time of that batch. The ready time of a batch is the latest ready time of all the jobs clustered in a batch. Term Q_2 is the chosen constant as it is sufficiently large in value to satisfy $y_{bb'k} = 0$ or 1 which is required for constraints (4-9)-(4-14). Constraint (4-9) ensures the satisfaction of the inequality in $t_{bk} + pt_b \leq t_{b'k}$, if batch b precedes batch b' ($y_{bb'k} = 1$). Constraints (4-10)-(4-14) are the precedence constraints provided by Pearn *et al.* [67]. Constraints (4-10) and (4-11) guarantee that one batch should precede another ($y_{bb'k} + y_{b'bk} = 1$) if two batches are scheduled on the same machine ($z_{bk} + z_{b'k} - 2 = 0$). It should be noted that the precedent relationships ($y_{bb'k}$) between batches b and b' on machine m_k may not be limited to direct ones. Constraint (4-12) ensures that the precedence variables $y_{bb'k}$ and $y_{b'bk}$ should be set to zero ($y_{bb'k} + y_{b'bk} \leq 0$) if any two batches b and b' are not scheduled on the machine m_k ($z_{bk} + z_{b'k} = 0$). Constraints (4-13) indicates the situation in which batch b is scheduled on machine m_k and batch b' is scheduled on another machine ($z_{b'k} - z_{bk} + 1 = 0$) and constraint (4-14) indicates the situation in which batch b' is scheduled on machine m_k and batch b is scheduled on another machine ($z_{bk} - z_{b'k} + 1 = 0$). Constraints (4-15)-(4-17) indicate that x_{jbk}, $y_{bb'k}$, and z_{bk} are binary integer variables. Finally, constraint (4-18) indicates that the makespan is greater than or equal to zero. The total number of variables is

$NBK + B^2K + BK + 1$ and the total number of constraint equations is $(9/2)B^2K + 3NBK - (3/2)BK + N + 2B + 1$, where B is the number of batches.

To demonstrate the applicability of Model P, an illustrative example is considered. The example involves two parallel batch processing machines (m_1 and m_2) and seven independent jobs with various sizes and processing times, which are ready for different starting times, as shown in Table 4-1. The seven jobs should be clustered into an appropriate number of batches. Those batches are scheduled on the two identical machines. The batch processing time is not affected by the machine processing it, but is dependent on the batch formation. The maximum number of pieces of one batch in a machine is set at 450 pieces in this example.

Table 4-1 Job sizes, ready times, and processing times of the seven independent jobs.

Job ID	size (pieces)	ready time	processing time
1	50	6	160
2	200	40	120
3	240	8	90
4	180	10	190
5	400	80	290
6	300	30	160
7	150	80	200

Model P is implemented using the software CPLEX OPL 3.5 to solve the seven-job example. For the example investigated, the model contains 225 variables and 736 equations. The MILP model is run on a Pentium IV 3.2GHz PC to obtain optimal solutions. The four batches, b_1, b_2, b_3, and b_4, are actually formed and their batch processing times are 90, 190, 200, and 290, respectively. Job 3 is grouped into b_1 and scheduled on Machine 1. Jobs 1, 2, and 4 are grouped into b_2 and scheduled on Machine 2. Moreover, jobs 6 and 7 are grouped into b_3 which is scheduled on Machine 2 and processed after b_2. Job 5 is the only one in b_4 and it is scheduled on Machine 1

and processed after b_1. The makespan of the example is 430 as shown in Figure 4-3 and the computational time is 1041.3 CPU seconds.

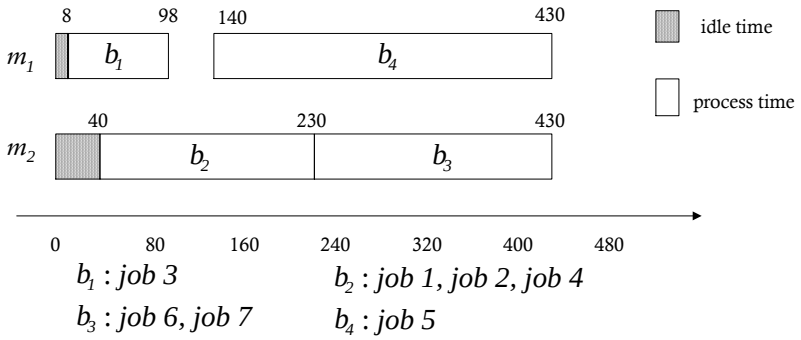


Figure 4-3 An optimal solution for the 7-job example with two aging test machines.

4.3. Compound MILP-based Algorithm

A compound MILP-based algorithm (*CMA*) improves the efficiency of the MILP model proposed in Section 4.2. For the aging test scheduling problem investigated in this dissertation, the number of batches and the composite jobs for each batch determine the solution quality and efficiency. It is obvious that the probable numbers of batches fall into the range, $1 \le B \le N$, in an *N*-job scheduling problem. The lower and upper bound values of B are obtained when all the N jobs are combined with one batch and each job is contained in an individual batch, respectively. However, exploring all the possible numbers of batches would increase the run time to obtain the optimal solution. Therefore, a relaxed MILP model (Model N) focuses on the batch formation and relaxes all relative precedence constraints to obtain the lower bound of the number of batches. Model N is then provided to reduce the search space for the aging test scheduling problem and is applied in the *CMA*. In Model N, the precedence variable ($y_{bb'k}$) and precedence constraints (constraint (4-9)–(4-14)) are removed. Model N uses

$\sum_{b=1}^{B}\sum_{k=1}^{K} z_{bk}$ as its objective function. Model N starts from a makespan lower bound $C'_{\max}$, which is greater than the longest processing time of all the jobs.

In this section, the compound MILP-based algorithm (*CMA*) is developed. The algorithm essentially consists of two phases. Phase I applies Model N to obtain the lower bound of the number of batches, which can serve as a referenced batch number in Model P. As mentioned earlier, it is sometimes advantageous to assign one more batch than the referenced batch number obtained from Phase I of the *CMA* to avoid excessive delays in waiting for the next scheduled late ready time job. Therefore, in Phase II, the solution of the subsequent batch number is checked. The algorithm is stated as follows and the flow chart is depicted in Figure 4-4.

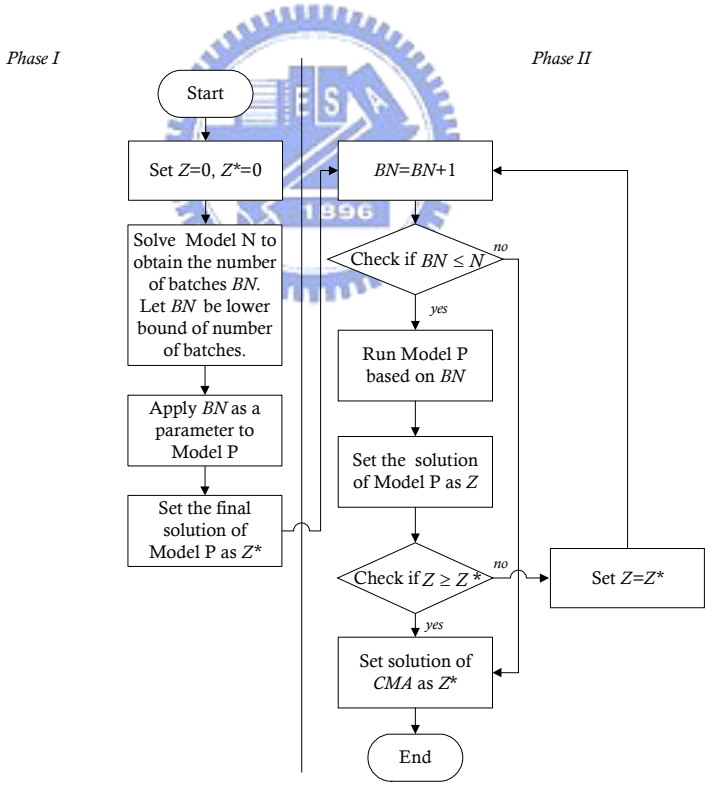


Figure 4-4 The flow chart of the compound MILP-based algorithm (*CMA*).

***Phase I:* The number of batches is determined and applied to Model P**

Step 1. Solve the following relaxed MILP model (Model N) in order to allow

the number of batches formed in the optimal solution to serve as lower bound batch numbers.

Model N:

$$\text{Minimize} \quad \sum_{b=1}^{B}\sum_{k=1}^{K} z_{bk} \tag{4-19}$$

subject to

constraints (4-2) –(4-8), and (4-15),

$$C'_{\max} \geq \max\{p_j\}. \tag{4-20}$$

Step 2. Denote the number of lower-bound batches to be the candidate number and denote it as BN. Apply BN as one parameter to Model P developed in Section 4.2. Denote the solution obtained within the limited computational time as *Z**.

Phase II: **The solution of subsequent batch number is checked**

Step 1. $BN = BN + 1$.

Step 2. If $BN \leq N$, then the batch number BN is applied as one parameter to Model P. Let the solution obtained from Model P be denoted as Z and go to Step 3. If the $BN > N$, then let Z^* be the final solution and stop the algorithm.

Step 3. If $Z \geq Z^*$, then let Z^* be the final solution and stop the algorithm. If $Z < Z^*$, then set Z as the new Z^* and go back to Step 1.

It should be noted that the computational complexity of the original MILP model (Model P) developed in Section 4.2 is reduced when the number of lower-bound batches is obtained by using the algorithm of Phase I of the *CMA*. The greater the difference between the number of jobs and the number of batches, the greater is the reduction in complexity. For the example with 7 jobs described in Section 4.2, the number of batches

is four, and it is obtained by using the algorithm in Phase I of the *CMA*. The composite jobs for these batches are presented in Table 4-2.

Table 4-2 The composite jobs for each estimated batch.

Batch name	Job ID	Batch size
b_2	1, 3, 7	440
b_3	5	400
b_6	6	300
b_7	2, 4	380

Using four batches as the parameter for Model P, 97 variables and 316 constraint equations are obtained. The solution is a makespan of 430 and is obtained within 1.3 CPU seconds by using CPLEX OPL 3.5. Moreover, when the batch number is set to five using Step 1 of Phase II of the *CMA*, the makespan is also 430 but the computational time is 10 CPU seconds. Therefore, the final solution of the example with seven jobs using the *CMA* is 430.

4.4. Heuristic Algorithms for Large-scale Problems

For small and moderate size aging test scheduling problems with unequal ready times and non-identical job sizes, the mixed integer linear programming model can provide optimal solutions within reasonable amounts of computational time. However, for large size aging test scheduling problems, solving the mathematical model is computationally inefficient. Therefore, three heuristic algorithms are proposed to generate efficient solutions for large problems. The proposed algorithms incorporate the merits of the DELAY heuristic solution procedure proposed by Lee and Uzsoy [51] with some modifications in order to accommodate the parallel batch processing machine environment. The DELAY heuristic algorithm allows for postponement in processing a batch in order to accommodate a job that is due to arrive soon and which might be combined with the delayed batch. This strategy can be used to avoid unacceptable delays

in the completion of scheduled batches. At other times, however, if the expected job is due to arrive much later, it is not advantageous to delay processing as this would cause excessive delays to jobs already waiting to be processed.

The three new heuristic algorithms include the characteristics of unequal ready times, non-identical job sizes, and parallel batch processing machines; they are referred to as Heuristic Algorithm 1 (*H1*), Heuristic Algorithm 2 (*H2*), and Mixed-strategy Heuristic Algorithm (*MixedH*). The first two heuristic algorithms essentially consist of two phases. Phase I of each modifies the DELAY algorithm (proposed by Lee and Uzsoy [51]) to form appropriate batches by adding a machine capacity checking step in order to accommodate the constraint of non-identical job sizes to enable the processing of batches of varied numbers of jobs. Furthermore, in Phase II of the two algorithms, the original single-machine scheduling idea proposed by Lee and Uzsoy [51] is also extended to accommodate a parallel batch processing machine environment. In this phase, the formed batches are then assigned to parallel batch processing machines and sequenced according to the batch ready times and batch processing times with a minimal makespan criterion. Finally, a mixed-strategy approach is also proposed. Algorithm details are presented as follows.

4.4.1. Heuristic Algorithm 1 (H1)

Phase I: *Batch formation*– the modified DELAY algorithm (Lee and Uzsoy [51])

Step 0. Let the available set be the set of jobs which are available to be selected as a batch. Sort all jobs associated with ready times in ascending order of magnitude as an unscheduled-job list. Index the ready times in the list as r_i. Assign the first job on the

unscheduled-job list into the available set. Set the decision time point (*t*) as the ready time of the first job (r_1) in the available set and set $i=1$ and $b=1$.

Step 1. Arrange the jobs associated with the processing times in the available set in descending order of magnitude. Choose the required number of early jobs in the available set which satisfy the constraint of machine capacity and place them in the candidate batch (*b*). The longest processing time of all the jobs in batch b serves as the corresponding batch processing time and is denoted as pt_b.

Step 2. Check whether there is a job (*j*) on the unscheduled-job list which satisfies constraints $r_j \le t + \alpha pt_b$ and $p_j \ge \alpha pt_b$, where $0 \le \alpha \le 1$. If a job (*j*) satisfies the conditions, then put job (*j*) into the candidate batch (b) and go to Step 3. Otherwise go to Step 4.

Step 3. If $\sum_{q \in b} p_q \le \beta \eta pt_b$ or $\sum_{q \in b} s_q > S'$ where $0 \le \beta \le 3$ and $\eta = \left\lceil \sum_{j=1}^{N} s_j / S' \right\rceil$, then they do not form a part of the candidate batch (b); therefore, let $i = i+1$, decision time point (*t*) be r_i, and then go to Step 5. Otherwise, go to Step4.

Step 4. Form candidate batch (b), let $t = t + pt_b$, then remove those jobs which are formed in batch b from the unscheduled-job list, set $i=1$, $b = b+1$, and go to Step 6.

Step 5. Select the maximum time value between the current decision time point and the smallest ready time in the unscheduled-job list as the new decision time point. Choose the jobs which have ready times which are earlier than the new point in decision time (*t*) to update the available set. Go back to Step 1 to reform the candidate batch.

Step 6. Repeat Step 5 until no more candidate jobs can be found on the unscheduled-job list.

Phase II: *Batch scheduling*

Step 1. Calculate the batch ready time and batch process time for each batch.

Step 2. Sort the batch ready times in ascending order of magnitude.

Step 3. If the ready times of some of the batches are equal, then sort the batch processing times in descending order of magnitude.

Step 4. Schedule the first batch into the first available machine and then remove the batch from the batch list.

Step 5. Repeat Step 4 of Phase II until all batches are scheduled.

In Step 2 of Phase I, Lee and Uzsoy [51] used parameter α to accommodate the postponement idea of the DELAY heuristic algorithm. Parameter α can be used to examine whether there exists a job with a ready time which is less than or equal to the summation of the current decision time point and the αpt_b time units. In addition, its corresponding job processing time is greater than or equal to the αpt_b. If such a job exists, the job might be combined with the delayed batch to avoid delaying that job.

4.4.2. Heuristic Algorithm 2 (H2)

In Phase II of Heuristic Algorithm 1 (*H1*), the batches are mainly scheduled based on their batch ready times. If the number of batches to be processed is slightly greater than the number of machines available to process them and, furthermore, if the processing times of some batches are relatively short, then the result may be that *H1* performs poorly because the individual short-process-time batches have been assigned to different machines resulting in a longer makespan. In attempting to improve the solution,

an alternative approach which combines the ready and processing times is proposed. The algorithm is described in following.

Phase I: *Batch formation*

Use the same process as for Phase I of Heuristic Algorithm 1 (*H1*).

Phase II: *Batch scheduling*

Step 1. Calculate the batch ready times (r_b') and the batch process times (pt_b), which are determined by the latest ready time and the longest processing time of all the jobs in one batch, respectively.

Step 2. Calculate $T_b = r_b' + pt_b$.

Step 3. Assign batches on K parallel batch processing machines using the longest processing time (LPT) rule (here it is assumed that the value of T_b represents the processing time for the LPT rule).

Step 4. Sequence batches on each machine in ascending order based on batch ready times.

4.4.3. Mixed-strategy Heuristic Algorithm (MixedH)

In attempting to obtain better solutions, a mixed-strategy approach, which has been considered by Frederickson [33], is presented. The mixed-strategy approach first uses *H1* and *H2* algorithms to generate two complete aging test scheduling problem solutions, then selects the best one. This approach is referred to as the Mixed-strategy Heuristic Algorithm (*MixedH*). As *MixedH* consists of two efficient algorithms, each of which performs some certain cases well, the solutions obtained from this algorithm are

improved. *MixedH* also involves two phases. Phase I, as in Phase I of *H1*, determines the batch formations. However, Phase II performs the solution procedures of the Phase II of *H1* and *H2* simultaneously to determine the final sequence and to select the value with the minimum makespan from the two solutions available.

4.5. Computational Results and Comparisons

For the purposes of testing the proposed algorithms and comparing them with the MILP model, an experiment involving two computational tests was designed to generate a series of problem instances. One computational test performed with small to moderate size problems obtains the solution quality of the proposed heuristic algorithms by comparing their solutions with the optimal solutions generated by the MILP model (Model P). The other computational test involves the large size problem taken from a module assembly process at a TFT-LCD factory in Hsinchu Science-based Industrial Park, Taiwan. In the following, two computational results are provided.

4.5.1. Analysis of Results from Small and Moderate Size Problems

The experimental design involves two essential characteristics, ready time variation and processing time variation. These two variations are characterized by two magnitudes, large (L) and small (S). Accordingly, the ready times in a problem are generated from uniform distributions in [0,300], [0,100] for large and small variations, respectively. The processing times are generated from uniform distributions in [90,300], [100,200] for large and small variations, respectively. The structure and data of the test problems are generated covering a wide variety of scheduling problems encountered in industrial practice. A number of machines and a number of jobs are alternated in order to get twenty-four problem configurations. For each problem configuration, five problem

instances are randomly generated. Thus, 120 different problem instances are generated which are either small or moderate in size. The four different experimental factors are listed in Table 4-3. The aging oven can process a batch in which the total number of pieces from all the jobs in that batch does not exceed 450 pieces. Without loss of generality, we assume that the size of each job is less than the machine capacity (i.e. 450 pieces of panel). Once the batch processing begins, it is non-preemptive until the batch is completely processed. Processing and ready times are measured in minutes. All jobs should be formed as batches and be processed completely by the minimum makespan.

Table 4-3 Experimental factors for small and moderate sized problems.

Factor	Value considered	Number of values
Number of jobs (*N*)	7, 15, 20	3
Ready time variation	L, S	2
Processing time variation	L, S	2
Number of machines (*K*)	2, 3	2
Total problem configurations		24
Instances per configuration		5
Total problem instances		120

Table 4-4 presents the solutions generated by the all the proposed algorithms on the eight small problem configurations with seven jobs in each. The values of the optimal solutions are obtained by solving the MILP model (Model P), which is formulated in Section 4.2. In Table 4-4, the problem configuration "7LL2" represents the 7 jobs with large ready time and large processing time variations, which are processed on two batch machines. In this testing, the run times of Model P and the *CMA* may vary for problem instances with different configurations. However, the run times of the *CMA* are significantly faster than for the original MILP model (Model P). All the solutions of the *CMA* are equal to the values obtained from Model P; hence the solutions are optimal. In the three heuristic algorithms, their performances are sensitive to the values of the

parameters α and β (as concluded also by Lee and Uzsoy [51]). This section provides the experiments involving the three heuristic algorithms to the testing problem instances using several values of α, which are initially set to 0, 0.2, 0.4, 0.6, 0.8, and 1 ($0 \le \alpha \le 1$) and β, which are initially set to 0, 0.2, 0.4, …, 2.6, 2.8, and 3 ($0 \le \beta \le 3$). The best solution, obtained using one of the parameter combinations, is selected as the final solution to the heuristic algorithm. It is worthwhile to note that the *MixedH* obtains 34 (out of 40) optimal solutions within 0.8 CPU seconds for each problem instance.

Table 4-5 displays the results for the problem instances with fifteen jobs and different configurations and their performance comparisons in terms of the makespan obtained using mathematical and heuristic algorithmic solutions. In the *MILP* model (Model P) and Model N of the *CMA*, the depth-first search strategy (Wolsey [93]) is implemented by choosing the most recently created node. To avoid the CPLEX routine which requires a tremendous amount of computation time, the maximum run time is set at 28800 CPU seconds. Furthermore, the nodes created cannot be greater than 1E06 in Phase II of the *CMA* in order to check the subsequent batch numbers repeatedly. CPLEX could stop at the pre-determined time without guaranteeing optimality for problems with high computational complexity. However, the depth-first search strategy can incorporate the strong branching rule (Wolsey [93]) causing the variable selection based on partially solving a number of sub-problems with tentative branches in order to find the most promising branch. Table 4-5 shows that the performances of the *CMA* are reasonably good; that is, with all the problem instances it achieved better solutions than the original MILP model (Model P). Furthermore, the three heuristic algorithms perform well and

efficiently. As seen in Table 4-5, due to its solution strategy, the *MixedH* provides the best solutions of all three heuristic algorithms for all problem configurations.

Furthermore, table 4-6 displays the results for the problem instances with twenty jobs and different configurations and their performance comparisons in terms of the makespan obtained using mathematical and heuristic algorithmic solutions. In table 4-6, the run times of *MILP* model (Model P) and Model N of the *CMA* almost approach to 28800 seconds, which is the maximum run times allowed. Therefore, these mathematical methods, including *MILP* and *CMA* models, are both computationally inefficient. Notably, the *MixedH* receives 15 better solutions (out of 40) than those of the two mathematical based algorithms. Thus, as the number of job increases to exceed 20 jobs, the *MixedH* not only runs fast but also obtains satisfactory solutions.

Table 4-7 displays the performance comparisons among the five algorithms in terms of (1) average rank, (2) average run times, (3) number of problem instances receiving the best solutions, and (4) number of optimal solutions. The results indicate that the *CMA* can obtain 40 (out of 40) optimal solutions for the 7-job test problem instances and it can perform remarkably well for the 15-job and 20-job problem instances. The *CMA* significantly speeds up the original MILP model in test problem instances. However, when the number of jobs is increased to 20 jobs, the computation times required by *CMA* are also increased and inefficient. It should also be noted that all three heuristic algorithms run very fast. With the 120 problem instances tested, it was found that none of them required more than 2 CPU seconds on a Pentium IV 3.2GH$_Z$ PC. To access the accuracy of the heuristic solutions, the best solution selected from the *CMA* and the

Table 4-4 Run times and makespan results for 7-job problem instances.

	Prob. Config.	*MILP*		*CMA*		*H1*		*H2*		*MixedH*	
		$C_{\max}$	Time (sec)	$C_{\max}$	Time (sec)	$C_{\max}$	Time (sec)	$C_{\max}$	Time (sec)	$C_{\max}$	Time (sec)
1	7LL2	459	2496	459	11.2	459	0.672	486	0.641	459	0.688
	7LL3	379	8742	379	71.9	379	0.625	379	0.625	379	0.656
	7LS2	395	501.2	395	13.7	395	0.609	401	0.609	395	0.672
	7LS3	345	1032	345	21.4	365	0.625	345	0.625	345	0.656
	7SL2	430	1041.3	430	11.2	430	0.656	480	0.625	430	0.676
	7SL3	370	648	370	1.7	370	0.625	370	0.688	370	0.719
	7SS2	346	1268	346	12.1	346	0.625	393	0.625	346	0.656
	7SS3	289	15475	289	66.3	289	0.609	306	0.625	289	0.672
2	7LL2	607	463	607	5.5	607	0.613	631	0.625	607	0.656
	7LL3	540	15.8	540	6.2	568	0.609	540	0.609	540	0.656
	7LS2	488	482	488	6.4	488	0.625	502	0.672	488	0.712
	7LS3	418	2899.8	418	32.6	418	0.625	422	0.625	418	0.672
	7SL2	561	450	561	18.3	570	0.641	575	0.609	570	0.719
	7SL3	435	28052	435	150.3	435	0.609	435	0.609	435	0.656
	7SS2	348	1376	348	25.2	348	0.641	367	0.703	348	0.756
	7SS3	267	20714	267	115.9	267	0.625	267	0.609	267	0.688
3	7LL2	522	1.6	522	1.1	522	0.625	599	0.609	522	0.734
	7LL3	522	7.5	522	5.3	522	0.641	522	0.594	522	0.672
	7LS2	455	1.3	455	1.1	455	0.609	471	0.594	455	0.672
	7LS3	455	10	455	3	455	0.641	455	0.609	455	0.672
	7SL2	424	693.3	424	8.7	424	0.641	468	0.609	424	0.734
	7SL3	332	7389.2	332	4.4	358	0.625	332	0.672	332	0.692
	7SS2	330	461.8	330	11	345	0.625	343	0.594	343	0.656
	7SS3	303	23341	303	85	318	0.625	303	0.609	303	0.813
4	7LL2	630	1606	630	13.1	656	0.609	656	0.609	656	0.672
	7LL3	543	41721	543	64.8	619	0.625	543	0.625	543	0.672
	7LS2	411	7.7	411	2.9	411	0.625	530	0.594	411	0.656
	7LS3	411	12.4	411	6.8	411	0.672	411	0.609	411	0.734
	7SL2	512	4209.9	512	7.7	512	0.625	534	0.672	512	0.756
	7SL3	425	31346.8	425	117.8	465	0.625	425	0.703	425	0.741
	7SS2	313	462.8	313	9.6	317	0.625	336	0.609	317	0.641
	7SS3	258	22398	258	61.6	261	0.625	258	0.609	258	0.641
5	7LL2	574	1306.8	574	6.4	592	0.641	590	0.641	590	0.656
	7LL3	555	7.6	555	5.3	555	0.625	555	0.609	555	0.655
	7LS2	453	1.6	453	0.7	454	0.625	453	0.703	453	0.741
	7LS3	453	10.9	453	4.8	453	0.609	453	0.719	453	0.741
	7SL2	498	3310	498	15.6	498	0.625	534	0.609	498	0.766
	7SL3	398	43339	398	134.2	398	0.719	398	0.609	398	0.752
	7SS2	371	3748	371	23.1	393	0.609	373	0.672	373	0.741
	7SS3	294	22360	294	96.1	298	0.625	294	0.609	294	0.719

The underlined values represent the best solutions for each problem instance from among all of the algorithms.

Table 4-5 Run times and makespan results for 15-job problem instances.

	Prob. Config.	*MILP*		*CMA*		*H1*		*H2*		*MixedH*	
		$C_{\max}$	Time (sec)	$C_{\max}$	Time (sec)	$C_{\max}$	Time (sec)	$C_{\max}$	Time (sec)	$C_{\max}$	Time (sec)
1	15LL2	592	28800	579	786.8	610	0.844	612	0.797	610	0.947
	15LL3	552	28800	552	15.3	552	0.781	552	0.781	552	0.931
	15LS2	467	250.6	467	5.1	467	0.781	467	0.766	467	0.916
	15LS3	467	7471.3	467	19.8	467	0.797	467	1.031	467	1.041
	15SL2	532	28800	532	1096.2	532	0.813	575	0.766	532	0.994
	15SL3	382	28800	382	2789.5	382	0.875	382	0.875	382	0.947
	15SS2	372	28800	372	1252.5	409	0.828	392	0.766	392	0.931
	15SS3	276	28800	275	2225	303	0.938	275	0.766	275	0.994
2	15LL2	614	28800	605	3756.1	629	0.813	713	0.766	629	0.978
	15LL3	565	642	565	17.4	565	0.797	565	0.750	565	0.947
	15LS2	498	28800	498	3063	516	0.797	551	0.797	516	0.931
	15LS3	478	28800	478	31.6	478	0.813	478	0.750	478	0.931
	15SL2	500	28800	475	1085.9	475	0.813	537	0.750	475	0.963
	15SL3	380	28800	380	150.7	380	0.797	380	0.750	380	0.916
	15SS2	380	28800	374	8478	380	0.781	403	0.766	380	0.947
	15SS3	293	28800	293	183.9	293	0.797	293	0.750	293	0.931
3	15LL2	505	28800	505	902.3	505	0.797	596	0.750	505	0.963
	15LL3	442	28800	442	48.2	442	0.781	442	0.750	442	0.916
	15LS2	455	28800	455	872.4	467	0.781	498	0.750	467	0.931
	15LS3	424	28800	424	18.8	426	0.875	426	0.813	426	0.963
	15SL2	425	28800	425	999	428	0.797	466	0.750	428	0.931
	15SL3	323	28800	314	1089	323	0.797	314	0.766	314	0.931
	15SS2	375	28800	375	1290	393	0.781	420	0.750	393	0.916
	15SS3	318	28800	306	3012.9	315	0.797	306	0.750	306	0.947
4	15LL2	495	28800	495	1185	516	0.797	546	0.750	516	0.931
	15LL3	445	28800	445	16.9	445	0.859	469	0.813	445	0.978
	15LS2	435	28800	422	997.7	435	0.781	456	0.828	435	0.947
	15LS3	370	28800	370	1275.5	375	0.781	390	0.734	375	0.892
	15SL2	459	28800	459	1323	473	0.797	473	0.750	473	0.910
	15SL3	369	28800	369	219	369	0.781	369	0.766	369	0.947
	15SS2	384	28800	383	1437	403	0.766	410	0.734	403	0.916
	15SS3	327	28800	327	3684	331	0.781	331	0.750	331	0.916
5	15LL2	553	28800	553	1014.7	586	0.828	590	0.766	586	0.963
	15LL3	538	813	538	13.1	538	0.797	538	0.781	538	0.916
	15LS2	454	28800	454	472.2	464	0.875	488	0.828	464	0.994
	15LS3	451	28800	451	22.7	451	0.859	451	0.828	451	0.993
	15SL2	517	28800	477	1483.9	505	0.797	517	0.750	505	0.916
	15SL3	363	28800	350	697.5	363	0.828	363	0.750	363	0.916
	15SS2	374	28800	368	3138	390	0.797	390	0.750	390	0.900
	15SS3	324	28800	321	4298	368	0.813	326	0.750	326	0.947

The underlined values represent the best solutions for each problem instance from among all of the algorithms.

Table 4-6 Run times and makespan results for 20-job problem instances.

	Prob. Config.	*MILP*		*CMA*		*H1*		*H2*		*MixedH*	
		$C_{\max}$	Time (sec)	$C_{\max}$	Time (sec)	$C_{\max}$	Time (sec)	$C_{\max}$	Time (sec)	$C_{\max}$	Time (sec)
1	20LL2	821	28800	757	28800	790	1.094	875	1.094	790	1.109
	20LL3	595	28800	573	774.2	601	1.125	625	1.078	601	1.250
	20LS2	656	28800	592	28800	575	1.063	635	1.031	575	1.141
	20LS3	498	28800	459	147.48	459	1.063	514	1.047	459	1.453
	20SL2	790	28800	682	28800	677	1.109	745	1.094	677	1.766
	20SL3	527	28800	489	28800	559	1.063	508	1.063	508	1.094
	20SS2	573	28800	529	28800	527	1.063	548	1.063	527	1.094
	20SS3	483	28800	380	28800	388	1.078	395	1.047	388	1.094
2	20LL2	939	28800	759	28800	792	1.078	918	1.109	792	1.188
	20LL3	663	28800	588	28800	609	1.078	613	1.078	609	1.094
	20LS2	613	28800	551	28800	566	1.078	618	1.063	566	1.109
	20LS3	455	28800	455	76.16	455	1.094	455	1.094	455	1.125
	20SL2	805	28800	744	28800	741	1.094	809	1.078	741	1.141
	20SL3	631	28800	544	28800	575	1.094	558	1.094	558	1.125
	20SS2	601	28800	519	28800	518	1.063	526	1.047	518	1.094
	20SS3	461	28800	371	28800	382	1.063	393	1.063	382	1.203
3	20LL2	1000	28800	876	28800	903	1.078	894	1.094	894	1.188
	20LL3	703	28800	622	28800	645	1.078	687	1.078	645	1.203
	20LS2	641	28800	568	28800	555	1.078	690	1.078	555	1.109
	20LS3	484	28800	471	28800	473	1.078	480	1.078	473	1.094
	20SL2	887	28800	793	28800	770	1.094	886	1.078	770	1.172
	20SL3	603	28800	579	28800	585	1.125	594	1.078	585	1.219
	20SS2	664	28800	521	28800	545	1.094	569	1.125	545	1.172
	20SS3	458	28800	383	28800	382	1.078	397	1.078	382	1.109
4	20LL2	925	28800	805	28800	776	1.109	948	1.109	776	1.141
	20LL3	671	28800	639	28800	656	1.141	680	1.125	656	1.188
	20LS2	604	28800	578	28800	598	1.094	670	1.078	598	1.141
	20LS3	478	28800	455	28800	456	1.078	507	1.063	456	1.219
	20SL2	826	28800	748	28800	765	1.094	822	1.094	765	1.109
	20SL3	611	28800	531	28800	513	1.094	570	1.063	513	1.109
	20SS2	632	28800	522	28800	514	1.078	580	1.047	514	1.109
	20SS3	450	28800	395	28800	410	1.078	414	1.094	410	1.109
5	20LL2	950	28800	775	28800	769	1.094	830	1.094	769	1.109
	20LL3	673	28800	578	28800	567	1.078	655	1.063	567	1.094
	20LS2	638	28800	549	28800	552	1.078	714	1.078	552	1.094
	20LS3	475	28800	443	967.66	443	1.078	475	1.109	443	1.156
	20SL2	880	28800	748	28800	693	1.109	816	1.078	693	1.397
	20SL3	643	28800	530	28800	490	1.094	559	1.063	490	1.344
	20SS2	577	28800	478	28800	495	1.078	524	1.063	495	1.297
	20SS3	432	28800	347	28800	354	1.078	374	1.109	354	1.266

The underlined values represent the best solutions for each problem instance from among all of the algorithms.

original MILP model (Model P) is used for each problem instance as a convenient reference point. The average percentage deviations between the *MixedH* and the selected best solutions of the two mathematical based algorithms are 0.36%, 1.8%, and 0.64% for 7-job, 15-job, and 20-job problem instances, respectively. The percentage deviation is defined as $(V_{MixedH} - V)/V$, where V_{MixedH} and V are the values for each problem instance which is obtained using the *MixedH* and the two mathematical based algorithms, respectively. The *CMA* may perform inefficiently as the number of job increases to the large scale usual in real-world factories. Thus, if the computational time is a primary concern, *MixedH* can solve real-world problems well.

Table 4-7 Comparisons of the five algorithms.

Problem		*MILP*	*CMA*	*H1*	*H2*	*MixedH*
7-job	Average rank among the five algorithms[a]	1	1	2.23	2.9	1.3
	Average run times (in CPU seconds)	7335.3	31.5	0.629	0.630	0.696
	Number of problem instances receiving the best solutions	40	40	25	19	34
	Number of optimal solutions	40	40	25	19	34
15-job	Average rank among the five algorithms[a]	1.6	1	2.3	3.1	1.95
	Average run times (in CPU seconds)	26149	1362	0.810	0.776	0.943
	Number of problem instances receiving the best solutions	26	40	15	15	19
	Number of optimal solutions	4	4	4	4	4
20-job	Average rank among the five algorithms[a]	4.55	1.7	1.75	4.025	1.575
	Average run times (in CPU seconds)	28800	25969	1.086	1.079	1.181
	Number of problem instances receiving the best solutions	1	25	18	1	18
	Number of optimal solutions	1	4	3	1	3

[a] The smallest rank value indicates the best solutions among all algorithms.

4.5.2. Analysis of the Result Based on the Large Scaled Problem

To demonstrate the applicability of the *MixedH* heuristic algorithm in real world problems, the following problem taken from a module assembly process of a TFT-LCD factory in Hsinchu Science-based Industrial Park, Taiwan, is considered. For the parallel batch processing machine scheduling problem in the aging test operation, there are 100 jobs (which can be categorized into 35 product families) to be processed on 6 identical

aging test machines arranged in parallel. The maximum batch size in one machine is 450 pieces of panel. It is assumed that the job testing recipes require the same testing temperature, which is normally set to a high temperature (55°C).

In order to solve the real-world aging test problem with unequal ready times and non-identical job sizes using the *MixedH* algorithm, the program codes of the three heuristic algorithms are written in Visual Basic 6.0. As a result, the *MixedH* algorithm runs quite efficiently. In fact, the *MixedH* algorithm takes 6.8 CPU seconds to obtain makespan 1330 on six aging test machines.

5. Conclusions and Future Research

5.1. Conclusions

The thin film transistor liquid crystal display (TFT-LCD) is a high competence industry. To be competitive, how to utilize their capacity and to meet customers' due dates to increase their profitability is essential. This dissertation tackles the production scheduling problems for the module assembly manufacturing process. According to the concept of theory of constraints (TOC), a printed circuit board bonding scheduling problem (PCBSP) is first investigated because it is often the cause of bottlenecks in the process and is mainly determined the performance of a system. To expedite the completion of the jobs, which have passed the PCB bonding operation, a batch processing machine scheduling problem, aging test scheduling problem (ATSP), is also solved because improper batch formation and scheduling may cause a large makespan.

The main contributions of this dissertation are to provide solution procedures of exact solutions and approximate solutions for the production scheduling problems (PCBSP and ATSP) in the module assembly manufacturing process. To obtain the exact solution of the PCBSP, we present a mixed integer linear programming model based on the MILP model proposed by Pearn *et al.* [67] with some modifications to accommodate the distinct objective function and constraints of the PCBSP. For the other essential scheduling problem in module assembly factories (ATSP), at the time this dissertation was being written, the proposed MILP model is the first one to obtain the exact solution for the research topic of the parallel batch processing machine scheduling problem with considerations of the compatible product families and unequal ready times. In the

proposed MILP model, the bi-functional objective is adopted to seek the minimal makespan and to incorporate the number of batches as the subsidiary in order to reduce the complexity of batch sequencing. This model can help the other researchers to obtain exact solutions for the similar batch scheduling problems successfully in their further investigations. Furthermore, an effective compound MILP-based algorithm (*CMA*) is also proposed to predetermine the number of batches in order to reduce the complexity associated with a search of all probable numbers of batches. From the computational tests conducted in this dissertation, the *CMA* significantly outperforms the original MILP model (Model P) within the limited computational time.

However, solving the MILP models to obtain exact solutions requires a tremendous computational time for large-scale problems. If the computational time is a primary concern, efficient solution procedures for the module assembly scheduling problems are also proposed based on the existing network algorithms and batch formation technologies. For the PCBSP, two network algorithms are investigated and two modifications are developed to solve this problem efficiently. To test the performances of those algorithms, a set of test problems was designed. The design of test problems involves four critical factors including the workload level of contract jobs, tightness of due date, setup time variation, and variation of contract/spot weight ratio. The computational test results reveal that the MPSA_TP perform stably in varied situations. All proposed algorithms solve the large-scale PCBSP effectively.

For the ATSP, three efficient heuristic algorithms are also developed for solving large-scale problems. The performances of the three heuristic algorithms are quite satisfactory. In particular, the *MixedH* shows its superiority with respect to run time and

solution quality, which are essential for real-world factories to schedule their batch processing machines. A real-world problem taken from a module assembly shop floor at a TFT-LCD factory is solved by using *MixedH* to obtain the near optimal solution within a few CPU seconds. We believe that the effective algorithms investigated in this dissertation may assist those involved in TFT-LCD factories to make judicious scheduling decisions.

5.2. Future Research

From this dissertation, three possible concerns might be useful in further research. The first concern involves solving the two scheduling problems using metaheuristic solution procedures, such as tabu search (TS), genetic algorithm (GA), and simulated annealing (SA), since metaheuristics have been widely applied in the investigations of scheduling problems. The metaheuristics can embed the solutions obtained in this dissertation as the initial solutions and then apply more general solution procedures that explore the solution space to identify good solutions. Moreover, the second concern is to solve another type of scheduling problem in TFT-LCD factories, such as a batch processing machine scheduling problem with the consideration of unequal ready times and multiple-batch operations in cell assembly process. Finally, the third concern is the need to minimize the maximum completion time involved in the consideration of setup times for the parallel batch processing machine scheduling problem with the characteristics of unequal ready times, setup times, and incompatible product families.

References

[1] Allahverdi, A., Gupta, J. N. D. and Aldowaisan, T., "A review of scheduling research involving setup considerations," *Omega-International Journal of Management Science*, 27, pp. 219- 239, 1999.

[2] Archetti, C., Hertz, A. and Speranza, M. G., "Metaheuristics for the team orienteering problem," *Journal of Heuristics,* 13, pp. 49- 76, 2007.

[3] Balas, E., "The prize collecting traveling salesman problem," *Networks,* 19, pp. 621- 636, 1989.

[4] Baptiste, P., Marek, C., Dürr, C., Jawor, W. and Vakhania, N., "Preemptive scheduling of equal-length jobs to maximize weighted throughput," *Operations Research Letters*, 32, pp. 258- 264, 2004.

[5] Bar-Yehuda, R., Even, G. and Shahar, S. M., "On approximating a geometric prize-collecting traveling salesman problem with time windows," *Journal of Algorithms,* 55, pp. 76- 92, 2005.

[6] Bilge, U., Kirac, F., Kurtulan, M. and Pekgun, P., "A Tabu search algorithm for parallel machine total tardiness problem," *Computers and Operations Research*, 31, pp. 397- 414, 2004.

[7] Bodin, L. D., Golden, B. L., Assad, A. A. and Ball, M. O., "Routing and scheduling of vehicles and crews: the state of the art," *Computers and Operations Research*, 10, pp. 63- 211, 1983.

[8] Bräysy, O. and Gendreau, M., "Vehicle routing problem with time windows, part I: route construction and local search algorithms," *Transportation Science*, 39, pp. 104- 118, 2005.

[9] Bräysy, O. and Gendreau, M., "Vehicle routing problem with time windows, part II: metaheuristics," *Transportation Science*, 39, pp. 119- 139, 2005.

[10] Butt, S. E. and Cavalier, T. M., "A heuristic for the multiple tour maximum collection problem," *Computers and Operations Research*, 21, pp. 101- 111, 1994.

[11] Butt, S. E. and Ryan, D. M., "An optimal solution procedure for the multiple tour maximum collection problem using column generation," *Computers and Operations Research*, 26, pp. 427- 441, 1999.

[12] Chandru, V., Lee, C. Y. and Uzsoy, R., "Minimizing total completion time on a batch processing machines with job families," *Operations Research Letters*, 13, pp. 61- 65, 1993.

[13] Chandru, V., Lee C. Y. and Uzsoy, R., "Minimizing total completion time on a batch processing machines," *International Journal of Production Research*, 31, pp. 2097- 2121, 1993.

[14] Chang, P. Y., Damodaran, P. and Melouk, S., "Minimizing makespan on parallel batch processing machines," *International Journal of Production Research*, 42, pp. 4211- 4220, 2004.

[15] Chang, P. C. and Wang, H. M., "A heuristic for a batch processing machine scheduled to minimise total completion time with non-identical job sizes," *International Journal of Advanced Manufacturing Technology*, 24, pp. 615- 620, 2004.

[16] Chao, I. M., Golden, B. L. and Wasil, E. A., "A fast and effective heuristic for the orienteering problem," *European Journal of Operational Research*, 88, pp. 475- 489, 1996.

[17] Chao, I. M., Golden, B. L. and Wasil, E. A., "The team orienteering problem," *European Journal of Operational Research*, 88, pp. 464- 474, 1996.

[18] Chao, Y. F., "The production planning for LCD cell forming process," thesis, Department of Industrial Engineering and Engineering Management in National Tsing Hua University, Taiwan, 2005.

[19] Cheng, T. C. E. and Sin, C. C. S., "A state-of-the-art review of parallel-machine scheduling research," *European Journal of Operational Research*, 47, pp. 271- 292, 1990.

[20] Chou, F. D., "A joint GA+DP approach for single burn-in oven scheduling problems with makespan criterion," *International Journal of Advanced Manufacturing Technology*, 35, pp. 587- 595, 2007.

[21] Chou, F. D., Chang, P. C. and Wang, H. M., "A hybrid genetic algorithm to minimize makespan for the single batch machine dynamic scheduling problem," *International Journal of Advanced Manufacturing Technology*, 31, pp. 350- 359, 2006.

[22] Chou, F. D. and Wang, H. M., "Scheduling for a single semiconductor batch-processing machine to minimize total weighted tardiness," *Journal of the Chinese Institute of Industrial Engineers*, 2007 (accepted).

[23] Christofides, N., Mingozzi, A., Toth, P. and Sandi, C., *Combinatorial Optimization*, John Wiley & Sons: New York, 1979.

[24] Chu, C., "A branch-and-bound algorithm to minimize total flow time with unequal release dates," *Naval Research Logistics*, 39, pp. 859- 875, 1992.

[25] Damodaran, P. and Srihari, K., "Mixed integer formulation to minimize makespan in a flow shop with batch processing machines," *Mathematical and Computer Modelling*, 40, pp. 1465- 1472, 2004.

[26] Deng, X., Feng, H., Zhang, P., Zhang Y. and Zhu, H., "Minimizing mean completion time in a batch processing system," *Algorithmica*, 38, pp. 513- 528, 2004.

[27] Deng, X., Feng, H., Li, G. and Shi, B., "A PTAS for semiconductor burn-in scheduling," *Journal of Combinatorial Optimization*, 9, pp. 5- 17, 2005.

[28] Dumas, Y., Desrosiers, J., Gelinas, E. and Solomon, M. M., "An optimal algorithm for the traveling salesman problem with time windows," *Operations Research*, 43, pp. 367- 371, 1995.

[29] DuPont, L. and Dhaenens-Flipo, C., "Minimizing the makespan on a batch machine with non-identical job sizes: an exact procedure," *Computers & Operations Research*, 29, pp. 807- 819, 2002.

[30] DuPont, L. and Ghazvini, F. J., "A branch and bound algorithm for minimizing mean flow time on a single batch processing machine," *International Journal of Industrial Engineering*, 4, pp.197- 203, 1997.

[31] Erramilli, V. and Mason, S. J., "Multiple orders per job compatible batch scheduling," *IEEE Transactions on Electronics Packaging Manufacturing*, 29 (4), pp. 285- 296, 2006.

[32] Feillet,D., Dejax, F. and Gendreau, M., "Traveling salesman problems with profits," *Transportation Science*, 39(2), pp. 188- 205, 2005.

[33] Frederickson, G. N., "Approximation algorithms for some postman problems," *Journal of the Association for Computing Machinery,* 26(3), pp. 538- 554, 1979.

[34] Fung, S. P. Y., Chin, F. Y. L. and Shen, H., "Online scheduling of unit jobs with bounded importance ratio," *International Journal of Foundations of Computer Science*, 16(3), pp. 581- 598, 2005.

[35] Gendreau, M., Hertz, A., Laporte, G. and Stan, M., "A generalized insertion heuristic for the traveling salesman problem with time windows," *Operations Research*, 46, pp. 330- 335, 1998.

[36] Gensch, D. H., "An industrial application of the traveling salesman's subtour problem," *AIIE Transactions*, 10, pp. 362-370, 1978.

[37] Ghazvini, F. J. and DuPont, L., "Minimizing mean flow times criteria on a single batch processing machine with non-identical jobs sizes," *International Journal of Production Economics*, 55, pp. 273- 280, 1998.

[38] Golden, B. L., "Evaluate a sequential vehicle routing algorithm," *AIIE Transactions*, 9, pp. 204-208, 1977.

[39] Golden, B. L., Levy, L. and Vohra, R., "The orienteering problem," *Naval Research Logistics*, 34, pp. 307-318, 1987.

[40] Golden, B. L., Wang, Q. and Lin, L., "A multifaceted heuristic for the orienteering problem," *Naval Research Logistics*, 35, pp. 359-366, 1988.

[41] Goldratt, E. M., *Theory of Constraints: What is This Thing Called the Theory of Constraints and How Should It be Implemented?,* New York: North River, 1990.

[42] Gupta, A. K. and Sivakumar, A. I., "Optimization of due-date objectives in scheduling semiconductor batch manufacturing," *International Journal of Machine Tools & Manufacturing*, 46, pp.1671- 1679, 2005.

[43] Gupta, J. N. D. and Ruiz-Torres, A. J., "Generating efficient schedules for identical parallel machines involving flow-time and tardy jobs," *European Journal of Operational Research*, 167, pp. 679- 695, 2005.

[44] Hiraishi, K., Levner, E., and Vlach, M., "Scheduling of parallel identical machines to maximize the weighted number of just-in-time jobs," *Computers and Operations Research*, 29, pp. 841- 848, 2002.

[45] Hu, Y. J., "Lot sizing scheduling for color filter production" thesis, Department of Industrial Engineering and Engineering Management in National Tsing Hua University, Taiwan, 2003.

[46] Hwang, T. K. and Chang, S. C., "Design of a lagrangian relaxation-based hierarchical production scheduling environment for semiconductor wafer fabrication," *IEEE Transactions on Robotics and Automation,* 19(4), pp. 566- 578, 2003.

[47] Jeong, B., Kim, S. W. and Lee, Y. J., "An assembly scheduler for TFT LCD manufacturing," *Computers & Industrial Engineering*, 41, pp. 37- 58, 2001.

[48] Jeong, B., Sim, S. B., Jeong, H. S. and Kim, S. W., "An available-to-promise system for TFT LCD manufacturing in supply chain," *Computers & Industrial Engineering*, 43, pp. 191- 212, 2002.

[49] Kashan, A. H., Karimi, B. and Jolai, F., "Effective hybrid genetic algorithm for minimizing makespan on a single-batch-processing machine with non-identical job sizes," *International Journal of Production Research*, 44, pp. 2337- 2360, 2006.

[50] Lai, C. L., "Lot sizing scheduling for color filter RGB process," thesis, Department of Industrial Engineering and Engineering Management in National Tsing Hua University, Taiwan, 2005.

[51] Lee, C. Y. and Uzsoy, R., "Minimizing makespan on a single batch processing machine with dynamic job arrivals," *International Journal of Production Research,* 37 (1), pp. 219- 236, 1999.

[52] Lee, C. Y., Uzsoy, R. and Martin-Vega L. A., "Efficient algorithms for scheduling semiconductor burn-in operations," *Operations Research*, 40 (4), pp. 764- 775, 1992.

[53] Lee, Y. H. and Lee, B., "Push-pull production planning of the re-entrant process," *International Journal of Advanced manufacturing technology,* 22, pp. 922- 931, 2003.

[54] Lenstra, J. K. and Rinnooy Kan, A. H. G., "Complexity of vehicle routing and scheduling problems," *Networks*, 11, pp. 221- 227, 1981.

[55] Li, C. L. and Lee, C. Y., "Scheduling with agreeable release times and due dates on a batch processing machine," *European Journal of Operational Research*, 96, pp. 564- 569, 1997.

[56] Lin, J. T. and Chen, Y. Y., "A multi-site supply network planning problem considering variable time buckets- A TFT-LCD industry case," *International Journal of Advanced manufacturing technology,* 33, pp. 1031- 1044, 2007.

[57] Lin, J. T., Chen T. L. and Chen, W. J., "Capacity and product mix planning problem for TFT array multi-plant," *Journal of the Chinese Institute of Industrial Engineers*, 24 (6), pp. 489- 504, 2007.

[58] Little, J., Murty, K., Sweeney, D. and Karel. C., "An algorithms for the traveling salesman problem," *Operations Research*, 11, pp. 972- 989, 1963.

[59] Mathirajan, M. and Sivakumar, A. I., "A literature review, classification and simple meta-analysis on scheduling of batch processors in semiconductor," *International Journal of Advanced manufacturing technology,* 29, pp. 990- 1001, 2006.

[60] Melouk, S., Damodaran, P. and Chang, P. Y., "Minimizing makespan for single machine batch processing with non-identical job sizes using simulated annealing," *International Journal of Production Economics*, 87, pp. 141- 147, 2004.

[61] Mokotoff, E., "Parallel machine scheduling problems: a survey," *Asia-Pacific Journal of Operational Research*, 18, pp. 193- 242, 2001.

[62] Mönch, L., Balasubramanian, H., Fowler, J. W. and Pfund, M. E., "Heuristic scheduling of jobs on parallel batch machines with incompatible job families and unequal ready times," *Computer & Operations Research,* 32, pp. 2731- 2750, 2005.

[63] Mönch, L., Unbehaun, R. and Choung, Y. I., "Minimizing earliness-tardiness on a single burn-in oven with a common due date and maximum allowable tardiness constraint," *OR Spectrum*, 28, pp.177- 198, 2006.

[64] Mönch, L. and Unbehaun, R., "Decomposition heuristics for minimizing earliness-tardiness on parallel burn-in ovens with a common due date," *Computer & Operations Research,* 34, pp. 3380- 3396, 2007.

[65] Pearn, W. L., Chung, S. H. and Yang, M. H., "The wafer probing scheduling problem (WPSP)," *Journal of the Operational Research Society*, 53, pp. 864-874, 2002.

[66] Pearn, W. L., Chung, S. H. and Yang, M. H., "A case study on the wafer probing scheduling problem," *Production Planning and Control*, 13(1), pp. 66- 75, 2002.

[67] Pearn, W. L., Chung, S. H. and Yang, M. H., "Minimizing the total machine workload for the wafer probing scheduling problem," *IIE Transactions*, 34, pp. 211-220, 2002.

[68] Pearn, W. L., Chung, S. H., Yang, M. H. and Chen, Y. H., "Algorithms for the wafer probing scheduling problem with sequence-dependent set-up time and due date restrictions," *Journal of the Operational Research Society*, 55, pp. 1194-1207, 2004.

[69] Pearn, W. L., Chung, S. H., Chen, A. Y. and Yang, M. H., "A case study on the multistage IC final testing scheduling problem with reentry," *International Journal of Production Economics*, 88, pp. 257-267, 2004.

[70] Poon, C. K. and Zhang, P., "Minimizing makespan in batch machine scheduling," *Algorithmica*, 39, pp. 155- 174, 2004.

[71] Poon, C. K. and Yu, W., "On minimizing total completion time in batch machine scheduling," *International Journal of Foundations of Computer Science*, 15, pp. 593- 607, 2004.

[72] Poon, C. K. and Yu, W., "On-line scheduling algorithms for a batch machine with finite capacity," *Journal of Combinatorial Optimization*, 9, pp. 167- 186, 2005.

[73] Pott, C. N. and Kovalyov, M. Y., "Scheduling with batching: A review," *European Journal of Operational Research*, 120, pp. 228- 249, 2000.

[74] Rojanasoonthon, S., Bard, J. F. and Reddy, S. D., "Algorithms for parallel machine scheduling: a case study of the tracking and data relay satellite system," *Journal of the Operational Research Society*, 54, pp. 806- 821, 2003.

[75] Rosenkrantz, D. J., Stearns, R. E. and Lewis, P. M., "An analysis of several heuristics for the traveling salesman problem," *SIAM Journal on Computing*, 6, pp. 563-581, 1977.

[76] Sen, T., Sulek, J. and Dileepan, P., "Static scheduling to minimize weighted and unweighted tardiness: a state-of-the-art survey," *International Journal of Production Economics*, 83, pp. 1- 12, 2003.

[77] Shin, H. J. and Leon, V. J., "Scheduling with product family set-up times: an application in TFT LCD manufacturing," *International Journal of Production Research*, 42, pp. 4235- 4248, 2004.

[78] So, K. C., "Some heuristics for scheduling jobs on parallel machines with setups," *Management Science*, 36, pp. 467- 475, 1990.

[79] Solomon, M. M., "Algorithms for the vehicle routing and scheduling problems with time windows constraints," *Operations Research*, 35, pp. 254- 265, 1987.

[80] Su, C. T., Wang, P. S. and Hsu, C. C., "Effective Approaches for Low Temperature Polysilicon TFT LCD Post-Mapping Yield Control Problem," *IEEE Transactions on Automation Science and Engineering,* 2, pp. 198- 206, 2005.

[81] Su, C. T. and Yang, C. H., "Two-phased meta-heuristic methods for the post-mapping yield control problem," *International Journal of Production Research*, 44, pp. 4837- 4854, 2006.

[82] Sung, C. S. and Choung, Y. I., "Minimizing makespan on a single burn-in oven in semiconductor manufacturing," *European Journal of Operational Research*, 120, pp. 559- 574, 2000.

[83] Sung, C. S., Choung, Y. I., Hong, J. M. and Kim, Y. H., "Minimizing makespan on a single burn-in oven with job families and dynamic job arrivals," *Computer & Operations Research*, 29, pp. 995- 1007, 2002.

[84] Tang, H. and Miller-Hooks, E., "A tabu search heuristic for the team orienteering problem," *Computers and Operations Research*, 32, pp. 1379-1407, 2005.

[85] Tovia, F. S., Mason, J. and Ramasami, B., "A scheduling heuristic for maximizing wirebonder throughput," *IEEE Transactions on Electronics Packaging Manufacturing*, 27(2), pp. 145-150, 2004.

[86] Uzsoy, R., "Scheduling of a single batch processing machines with non-identical job sizes," *International Journal of Production Research*, 32, pp. 1615- 1635, 1994.

[87] Van Der Zee, D. J., "Dynamic scheduling of batch servers with compatible product families," *International Journal of Production Research*, 42, pp. 4803- 4826, 2004.

[88] Van Der Zee, D. J., "Dynamic scheduling of batch-processing machines with non-identical product sizes," *International Journal of Production Research*, 45, pp. 2327- 2349, 2007.

[89] Wang, C. S. and Uzsoy, R., "A genetic algorithm to minimize maximum lateness on a batch processing machine," *Computers & Operations Research*, 29, pp.1621- 1640, 2002.

[90] Wang, F. K., Du, T. and Wen, F. C., "Product mix in the TFT-LCD industry," *Production Planning & Control,* 18, pp. 584- 591, 2007.

[91] Wang, H. M., Chang, P. C. and Chou, F. D., "A hybrid forward/backward approach for single batch scheduling problems with non-identical job sizes," *Journal of the Chinese Institute of Industrial Engineers,* 24 (3), pp. 191- 199, 2007.

[92] Wang, P. S. and Su, T. C., "An optimal yield mapping approach for the small and medium sized liquid crystal displays," *International Journal of Advanced manufacturing technology,* 27, pp. 985- 989, 2006.

[93] Wolsey, L. A., *Integer Programming*, 1st ed. New York: John Wiley & Sons, 1998.

[94] Zhang, G., Cai, X., Lee C. Y. and Wong, C. K., "Minimizing makespan on a single batch processing machine with nonidentical job sizes," *Naval Research Logistics*, 48, pp. 226- 240, 2001.

Appendix

Table A1 Setup times matrix for 26 product types in problem 6 (unit: minutes).

To / From	U	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
U	0	120	120	120	120	120	120	120	120	120	120	120	120	120	120	120	120	120	120	120	120	120	120	120	120	120	120
1	0	0	15	20	20	360	360	180	30	30	180	360	250	250	180	30	30	150	120	50	50	50	130	20	50	30	20
2	0	15	0	20	20	360	320	180	30	30	180	360	30	30	170	170	250	30	50	50	150	100	100	150	15	250	30
3	0	20	20	0	120	30	30	170	170	250	250	250	130	150	50	260	80	30	50	50	150	100	100	150	15	80	30
4	0	280	280	360	0	130	20	50	260	80	80	280	130	150	50	260	80	30	50	50	150	100	100	150	15	80	30
5	0	280	280	360	15	0	20	50	260	80	80	280	130	150	50	260	80	30	50	50	150	100	100	150	15	80	30
6	0	280	280	360	15	20	0	50	260	80	80	280	15	80	150	150	120	80	250	250	250	100	100	270	30	120	80
7	0	20	20	120	50	15	80	0	150	120	120	30	15	80	150	150	120	100	150	150	150	80	80	30	30	120	100
8	0	20	20	15	320	30	120	120	0	250	250	250	50	150	150	260	80	250	170	170	250	250	250	250	30	80	20
9	0	280	280	200	50	50	150	150	260	0	80	280	250	100	250	150	30	250	50	260	80	80	80	280	30	30	20
10	0	150	150	80	80	250	250	250	270	30	0	20	150	80	280	30	250	15	170	130	250	250	250	250	150	30	50
11	0	270	270	100	100	150	150	280	30	30	100	0	50	180	120	120	250	150	170	130	250	250	250	250	270	50	150
12	0	15	15	30	30	180	180	120	120	50	150	150	0	180	120	120	250	280	250	250	100	30	30	280	250	250	280
13	0	20	20	360	360	260	80	80	280	250	250	100	50	0	120	150	150	15	360	360	320	320	120	15	280	150	15
14	0	280	280	15	15	260	80	80	280	150	150	80	20	15	0	150	208	15	320	320	320	320	120	15	280	30	15
15	0	20	20	50	50	130	250	250	250	150	80	80	150	80	80	0	80	250	50	260	80	80	80	280	30	250	250
16	0	20	20	208	208	280	250	30	30	280	280	280	280	100	30	30	0	130	150	150	120	120	120	30	280	50	150
17	0	280	280	50	50	150	150	80	80	120	250	280	150	80	80	80	80	0	20	20	20	50	50	150	280	250	250
18	0	150	150	80	80	270	270	100	100	50	150	280	270	100	100	100	100	15	0	130	250	50	250	250	150	150	150
19	0	30	50	30	30	15	280	120	120	30	30	15	15	280	280	120	120	30	280	0	250	250	100	30	15	180	180
20	0	80	250	30	280	15	15	120	120	30	30	15	15	360	360	120	120	30	280	280	0	250	100	30	150	30	30
21	0	100	150	30	280	15	50	250	250	80	80	250	250	360	360	120	120	250	250	100	30	0	30	30	30	30	30
22	0	150	150	80	150	50	50	150	150	100	100	150	15	20	20	250	250	250	250	100	30	30	0	250	30	170	250
23	0	270	270	100	150	50	50	50	180	120	120	180	15	20	20	250	250	250	250	100	30	30	30	0	260	260	80
24	0	30	50	30	30	360	280	120	120	30	30	15	250	280	280	120	120	30	280	280	250	250	100	30	0	180	180
25	0	80	250	30	280	360	360	120	120	30	30	15	250	280	280	120	120	30	280	280	250	250	100	30	150	0	180
26	0	80	250	30	280	360	360	120	120	30	30	15	250	280	280	120	120	30	280	280	250	250	100	30	150	0	0

Table A2 The job information of the 120 jobs in PCBSP.

Job ID	Product type	Processing time (min)	Weight	Due date	Contract/ Spot	Job ID	Product type	Processing time (min)	Weight	Due date	Contract/ Spot
1	1	213	50000	1440	C	61	12	213	75000	1440	C
2	1	213	50000	1440	C	62	12	213	75000	1440	C
3	1	213	50000	1440	C	63	12	213	65000	4320	S
4	1	213	40000	4320	S	64	12	213	65000	4320	S
5	1	213	40000	4320	S	65	13	183	72500	2880	C
6	2	192	60000	2880	C	66	13	183	72500	2880	C
7	2	192	60000	2880	C	67	13	183	72500	4320	C
8	3	213	63000	1440	C	68	13	183	62500	4320	S
9	3	213	63000	1440	C	69	13	183	62500	4320	S
10	3	213	63000	1440	C	70	14	213	65000	2880	C
11	3	213	53000	4320	S	71	15	183	60000	2880	C
12	3	213	53000	4320	S	72	15	183	60000	2880	C
13	4	175	50000	4320	C	73	15	183	60000	2880	C
14	5	183	59000	1440	C	74	15	183	60000	4320	C
15	5	183	59000	1440	C	75	15	183	50000	4320	S
16	5	183	59000	1440	C	76	15	183	50000	4320	S
17	5	183	49000	4320	S	77	15	183	50000	4320	S
18	5	183	49000	4320	S	78	16	175	78500	4320	C
19	5	183	49000	4320	S	79	16	175	78500	4320	C
20	5	183	49000	4320	S	80	16	175	78500	4320	C
21	6	213	76000	2880	C	81	16	175	78500	4320	C
22	6	213	76000	2880	C	82	16	175	68500	4320	S
23	6	213	76000	2880	C	83	17	213	60000	2880	C
24	6	213	76000	2880	C	84	17	213	60000	2880	C
25	6	213	76000	2880	C	85	18	183	54000	4320	C
26	6	213	66000	4320	S	86	18	183	54000	4320	C
27	6	213	66000	4320	S	87	18	183	44000	4320	S
28	6	213	66000	4320	S	88	18	183	44000	4320	S
29	6	213	66000	4320	S	89	19	200	70000	2880	C
30	6	213	66000	4320	S	90	19	200	70000	2880	C
31	7	183	70000	1440	C	91	20	200	60000	2880	C
32	7	183	70000	1440	C	92	20	200	60000	4320	C
33	7	183	70000	1440	C	93	20	200	60000	4320	C
34	7	183	60000	4320	S	94	20	200	60000	4320	C
35	7	183	60000	4320	S	95	20	200	50000	4320	S
36	7	183	60000	4320	S	96	21	192	50000	4320	C
37	7	183	60000	4320	S	97	21	192	50000	4320	C
38	8	167	66000	2880	C	98	21	192	50000	4320	C
39	8	167	66000	2880	C	99	21	192	50000	4320	C
40	8	167	66000	4320	C	100	21	192	50000	4320	C
41	8	167	66000	4320	C	101	21	192	50000	4320	C
42	8	167	66000	4320	C	102	22	213	62500	2880	C
43	9	192	75000	1440	C	103	22	213	62500	2880	C
44	9	192	75000	1440	C	104	23	213	58000	4320	S
45	9	192	75000	1440	C	105	23	213	58000	4320	S
46	9	192	65000	4320	S	106	23	213	58000	4320	S
47	9	192	65000	4320	S	107	24	200	71000	2880	C
48	9	192	65000	4320	S	108	24	200	71000	2880	C
49	9	192	65000	4320	S	109	24	200	71000	2880	C
50	10	183	60000	4320	C	110	24	200	71000	2880	C
51	10	183	60000	4320	C	111	25	183	61000	2880	C
52	10	183	50000	4320	S	112	25	183	61000	2880	C
53	10	183	50000	4320	S	113	25	183	61000	4320	C
54	11	167	50000	2880	C	114	25	183	61000	4320	C
55	11	167	50000	2880	C	115	25	183	51000	4320	S
56	11	167	40000	4320	S	116	25	183	51000	4320	S
57	11	167	40000	4320	S	117	26	192	65500	1440	C
58	11	167	40000	4320	S	118	26	192	55500	4320	S
59	12	213	75000	1440	C	119	26	192	55500	4320	S
60	12	213	75000	1440	C	120	26	192	55500	4320	S