

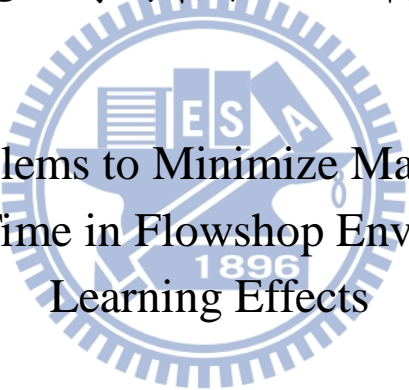
國立交通大學

工業工程與管理學系

博士論文

具有學習效果的流程式生產排程之最大完工時間與總
完工時間最小化之研究

Scheduling Problems to Minimize Makespan and Total
Completion Time in Flowshop Environment with
Learning Effects



研究生：鐘愉翔

指導教授：唐麗英 教授

洪瑞雲 教授

中華民國 一 百 年 十 一 月

具有學習效果的流程式生產排程之最大完工時間與總
完工時間最小化之研究

Scheduling Problems to Minimize Makespan and Total
Completion Time in Flowshop Environment with
Learning Effects

研 究 生：鐘愉翔

Student : Yu-Hsiang Chung

指導教授：唐麗英 教授

Advisor : Dr. Lee-Ing Tong

洪瑞雲 教授

Dr. Ruey-Yun Horng

國立交通大學
工業工程與管理學系



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

November 2011

Hsinchu, Taiwan, Republic of China

中華民國 一 百 年 十 一 月

具有學習效果的流程式生產排程之最大完工時間與總完工時間最小化之研究

研究生：鐘愉翔

指導教授：唐麗英 博士
洪瑞雲 博士

國立交通大學
工業工程與管理學系

摘 要

在傳統的排程問題中，工作件的加工時間皆設定為固定常數，且不會因工作件在機台上的加工順序而改變，然而當執行加工的人員因重複處理類似工作件而獲得經驗時，工作件的加工時間會因此縮減，此現象為近年來在排程領域被廣泛討論的「學習效果 (Learning effect)」。學習效果可分類為兩種型式，分別為「依加工順序改變之學習效果 (Position-based learning effect)」與「依已加工時間改變之學習效果 (Sum-of-processing-time-based learning effect)」，在一個排程問題中，此兩種型式之學習效果可以單獨考慮或同時考慮，由於依加工順序改變之學習效果模型為理論上的理想學習模型，因此本論文將探討依加工順序改變之學習效果。再者，目前關於具有學習效果的排程問題大多針對單機做探討，然而在實際製程上，許多生產環境的排程屬於多機流程式生產排程，其問題之求解的複雜性亦高於單機生產排程問題。此外，大多數排程問題之目的是求得最佳的工作件排序使其目標函數最小化，而最常被探討的目標函數為最大完工時間與總完工時間。因此本論文針對依加工順序改變之學習效果探討兩個多機流程式生產排程問題，第一個問題設定所有機台有相同的學習效果，其目標為最大完工時間最小化；第二個問題則是依不同機台有不同的學習效果，其目標為加權整合之總完工時間與最大完工時間最小化。

本論文對於工作件數少的問題，使用分枝界限演算法求得最佳排序，接著推導凌越性質與下界值以增進分枝界限演算法的求解效率；對於工作件數多的問題，本論文將利

用兩個知名的啟發式演算法、模擬退火法與基因演算法來求得近似解。最後，本論文將針對所有提出的演算法進行電腦模擬，以探討學習效果對於分枝界限演算法的求解效率與啟發式演算法的精準性之影響。針對本論文探討之問題的電腦模擬結果，我們發現將傳統環境中求得之最佳排序應用在具有學習效果的环境中，得到的目標函數值將比實際的最佳目標函數值大，此現象指出學習效果對本論文提出的排程問題有顯著的影響。而在求最佳排序時，分枝界限演算法求解的效率與學習效果的強度成正比。此外，基因演算法為本論文提出之求近似解的演算法中最精準的。基於分枝界限演算法求解時間的分佈變異大且呈右偏，因此本論文建議在求解時先施以分枝界限演算法，若在合理的時間內無法求得最佳排序，則施以基因演算法求近似排序。最後，若流程式生產環境中的機台上能指派不同的操作員，則將學習能力強的操作員指派至工作量較大的機台上能得到較佳的目標函數值。



關鍵字：流程式生產排程、學習效果、最大完工時間、總完工時間

Scheduling Problems to Minimize Makespan and Total Completion Time in Flowshop Environment with Learning Effects

Students: Yu-Hsiang Chung

Advisor: Dr. Lee-Ing Tong

Dr. Ruey-Yun Horng

Department of Industrial Engineering and Management,
National Chiao Tung University

ABSTRACT

In traditional scheduling problems, the processing time for a given job is assumed to be a fixed constant no matter the scheduling order of the job. However, it is noticeable that the job processing time declines as workers gain more experience. This phenomenon is called the “learning effect”. The learning effect is extensively studied in scheduling field recently, and it can be classified into two types: “the position-based learning” and “the sum-of-processing-time-based learning”. The two types of learning effect can be considered alone or simultaneously in a scheduling problem. The position-based learning is studied in this dissertation because of its model is the pure learning model in theory. In addition, most of the studies on the learning effect are focused only on single-machine setting. However, numerous real-world industrial problems belong to flowshop scheduling problems, and dealing with the flowshop scheduling problems is more complex than dealing with the single-machine problems. Most scheduling problems aim at determining an optimal sequence to minimize the objective function. The makespan and total completion time are the objective functions that are often studied. As a result, this dissertation discusses two flowshop scheduling problems with position-based learning effect. The learning effects are identical on all machines, and the purpose is to minimize the makespan in the first problem. The learning effects are distinct for different machines, and the purpose is to minimize the weighted sum of total completion time

and makespan in the second problem.

In this dissertation, the branch-and-bound algorithm is proposed to seek the optimal sequence for the small job-sized problem. Then the dominance properties and lower bounds are proposed to accelerate the procedure of the branch-and-bound algorithm. For the large job-sized problem, two well-known heuristic algorithms, simulated annealing and genetic algorithm are utilized to yield the near-optimal sequence. In the end, the simulated experiments are examined to assess the performance of the algorithms proposed in this dissertation. The computational results of the proposed problems reveal that the objective value calculated from the optimal sequence under the traditional environment is larger than the optimal objective value in the environment with learning considerations. It implies the influence of the learning effect is notable for the problems proposed in this dissertation. Furthermore, the efficiency of the branch-and-bound algorithm ascends as the learning effect enhances while seeking the optimal sequence. The proposed genetic algorithm has the best performance among all heuristic and meta-heuristic algorithms in terms of the accuracy. In addition, due to the large variance and the right skewness for the distribution of the execution time, the branch-and-bound algorithm is recommended to obtain the optimal sequence in a reasonable amount of time, or to derive the near-optimal sequence from the proposed genetic algorithm. Eventually, assigning the operator with stronger learning effect to the machine with heavier workload might derive smaller objective value while the operators are allocated in the flowshop environment.

Keywords: Flowshop scheduling; Learning effect; Makespan; Total completion time.

誌 謝

經歷四年的博士研究階段，愉翔首先要感謝 唐麗英 教授與 洪瑞雲 教授兩位老師無私的指導驚鈍的我作研究，有了兩位博學多聞的恩師寢囊相授，使本論文的内容更加嚴謹與充實。愉翔也從兩恩師身上學到了圓融的待人處事方法，對於老師四年中的照顧與教誨，愉翔將永遠銘記在心。

愉翔感謝論文口試委員 黎正中 教授、李榮貴 教授與 王春和 教授在本論文審查及口試其間的辛勞，並給予寶貴的建議及鼓勵，使愉翔能在未來的研究及生涯規劃上注入新的想法。

愉翔感謝祖父 鐘坤湖 先生、先祖母 鐘黃金女 居士、父親 鐘健倉 先生、母親 林媽媽 女士及妹妹 鐘樂珊 小姐對我不變的支持與鼓勵，讓我能無後顧之憂的專心作研究。另外也要感謝一直對愉翔抱持信心的親戚，大家的期許讓我更有努力向上的動力。

愉翔感謝碩士班的恩師 李文炯 教授與 吳進家 教授，兩位老師激發我對研究的熱誠，讓我有機會突破自我。最後要感謝 佳煌、元銘、凱斌學長、榮弘學長與一平學長，有了各位的陪伴與指導，使我四年的求學過程多采多姿。感謝所有幫助過我的人，愉翔會竭盡所能的努力，不辜負大家對我的期望。

鐘愉翔 謹誌

中華民國 一 百 年 十 一 月 于 交 通 大 學

Contents

Abstract (in Chinese)	i
Abstract (in English)	iii
Acknowledgements (in Chinese)	v
List of Tables	viii
List of Figures	ix
Chapter 1 Introduction	1
1.1 Research motivation	1
1.2 Literature review	3
1.3 Research objectives and methodologies	9
Chapter 2 Algorithms	12
2.1 Branch-and-bound algorithms	12
2.2 Heuristic algorithms	15
2.3 Meta-heuristic algorithms	17
Chapter 3 Makespan minimization for m-machine flowshop scheduling problem with position-based learning effects	22
3.1 Notations and problem statement	22
3.2 Dominance property	23
3.3 Lower bound	25
3.4 Computational results	28
3.5 Summary	38

Chapter 4 Bi-criteria minimization for m-machine flowshop scheduling problem with machine- and position-based learning effects	39
4.1 Notations and problem statement	39
4.2 Dominance property	40
4.3 Lower bound	42
4.4 Computational results	44
4.5 Summary	63
Chapter 5 Concluding remarks	64
5.1. Conclusion	64
5.2 Suggestions for further studies	65
References	66



List of Tables

Table 3.1. The influence of the learning effect on optimal solution	28
Table 3.2. The performance of the property and the lower bound for the branch-and-bound algorithm	29
Table 3.3. The performance of branch-and-bound algorithm and heuristic algorithms of different parameters	31
Table 3.4. The performance of branch-and-bound algorithm of different parameters after outliers elimination	34
Table 3.5. The relative percentage deviation of heuristic algorithms	36
Table 3.6. One-way ANOVA for <i>RPD</i> of four heuristics	37
Table 3.7. Tukey-test results of four heuristics	38
Table 4.1. The index set of the learning effects	46
Table 4.2. The performance of the branch-and-bound algorithm	49
Table 4.3. The comparison among five learning patterns for the optimal objective value	52
Table 4.4. The performance of the heuristic algorithms ($\alpha = 0.25$)	54
Table 4.5. The performance of the heuristic algorithms ($\alpha = 0.50$)	55
Table 4.6. The performance of the heuristic algorithms ($\alpha = 0.75$)	57
Table 4.7. Two-way ANOVA of the error percentages for all heuristic algorithms	59
Table 4.8. The comparison of the heuristic algorithms for large job-sized problem	62

List of Figures

Fig 2.1. The flowchart of the proposed branch-and-bound algorithm	14
Fig 2.2. The flowchart of the proposed genetic algorithm	21
Fig 3.1. Box-plot for logarithm scale with learning effect as 70%	32
Fig 3.2. Box-plot for logarithm scale with learning effect as 80%	33
Fig 3.3. Box-plot for logarithm scale with learning effect as 90%	33
Fig 4.1. The number of nodes for branch-and-bound algorithm under different α	47
Fig 4.2. The relative percentage deviation of the learning patterns for the optimal objective value under different α	48



Chapter 1

Introduction

In manufacturing and service industries, the scheduling problem is an important field of decision-making. In a narrow sense, the meaning of scheduling is to set the priorities of tasks for optimizing certain objectives. Due to the arising of global industrialization, many researchers and practitioners devote to study the scheduling problems, and the meaning of scheduling is extended to assign limited resources to the tasks for optimizing certain objectives. The resources of scheduling problems are manpower, raw materials, and facilities and so on. In fact, an inaccurate scheduling policy may lead to crucial loss of capacity or goodwill. Since the competition of marketplace grows rapidly, an effective scheduling policy plays a critical role for making profit for an enterprise.

1.1 Research motivation

In traditional scheduling problems, it is assumed that all the job processing times are fixed and known (Pinedo [29]; Smith [32]). However, job processing times frequently decline as workers gather working knowledge and experience. For example, processing similar tasks continuously improves worker's skills and helps them perform their jobs efficiently (Biskup [1]). This phenomenon is known as the "learning effect." The influence of learning on productivity for aircraft industry manufacturing was first observed by Wright [44], in which the processing time of a unit declines by 20% with every redoubling output and this phenomenon is called as 80% hypothesis. Afterward, the learning effect was affirmed in numerous industries such as the manufacturing and service industries (Yelle [48]).

The phenomenon of learning in Wright [44] is presented as $p_{\{k\}} = pk^a$, where $p_{\{k\}}$ denotes the actual processing time for each unit when the output requires k units, in which the

actual processing times of all units are identical; p denotes the normal processing time of a unit which is given before starting the process; and a denotes the learning index which is equal or less than zero and depended on the learning rate R . For the 80% hypothesis (i.e. $R = 0.8$), it is shown that $p_{\{2k\}} = 0.8p_{\{k\}}$, and it implies that $p(2k)^a = 0.8pk^a$, and then the learning index a is derived as $\log_2 0.8 = -0.322$. Therefore, the learning index a is set as $\log_2 R$ when the processing time of a unit decreases by $100(1 - R)\%$ with redoubling output. Subsequently, Biskup [1] applied the concept of Wright [44] and created a famous learning model by assuming every job is a unit even when the processing times are different among all jobs. Therefore, the actual processing time is based on the scheduled position of the job in the model proposed in Biskup [1].

In terms of the occurrence for the learning effect in the production activities, Biskup [2] stated that an inherent characteristic of the production environments with the learning effect is a high level of human activities, and these activities are presented as follows,

- All kinds of handcraft,
- Operating and controlling the machines,
- Setting up the machines,
- Maintaining the machine,
- Cleaning the machines,
- Removing the failure of the machines.

Hence, the learning effect occurs when the production activity belongs to the short-term planning. In addition, the learning effect also takes place if the production environments alter and some examples are presented as follows,

- Dealing with the jobs that have never been produced before,
- Hiring new employees,
- The procedure of the processes changes,

- Operating the equipments which are replaced or updated.

However, the influence of the learning should decline after a certain period of time because of the improvement for operator's skill is limited.

The learning effect has received significant attention in scheduling field recently. In the literature with regard to the learning effect, most studies focus on single-machine setting. The discussion of flowshop scheduling problems is rarely seen. Practically, in many manufacturing and assembly facilities, numbers of operations have to be done on every job and this production environment is modeled as flowshop. Therefore, in this dissertation, we intend to study the flowshop scheduling problems with learning effects.

1.2 Literature review

Biskup [1] is a pioneer to introduce a learning model into scheduling problems in which the actual processing time of a job decreases when the job is late scheduled. He examined the problems associated with minimizing the deviation from a common due date and the sum of flow times in a single-machine environment, and demonstrated that the problems are polynomially solvable. Subsequently, numerous studies have considered this novel and extended region. Cheng et al. [6] developed a model with learning effect in which actual job processing time is based on the total normal job processing time and the position of schedule on a single machine. They then demonstrated that the makespan and total completion time problems are polynomially solvable, and demonstrated that the problems for minimizing weighted completion time and maximum lateness are polynomially solvable with certain agreeable conditions. Janiak and Rudek [16] introduced a multi-ability learning effect into a makespan single-machine scheduling problem. They established polynomial time algorithms to optimize the special cases of the problem they proposed. Furthermore, Biskup [2] presented a detailed review of scheduling problems with learning effect. Particularly, he classified the

existing models into two distinct groups: the position-based learning and the sum-of-processing-time-based learning. The position-based learning is influenced by the number of jobs processed. Meanwhile, the sum-of-processing-time-based learning considers the processing time of the jobs processed to date.

In the position-based learning model, Lee et al. [23] studied a single-machine scheduling problem with release times under learning consideration. They proposed a branch-and-bound and a heuristic algorithm to obtain the optimal and near-optimal solution for minimizing the makespan. Zhu et al. [51] studied two single-machine group scheduling problems. The job processing time is a function of job position, group position and the amount of resources assigned to the group. They verified that minimizing the weighted sum of the makespan and the total resource cost remains polynomial solvable. Furthermore, Wang et al. [39] investigated a single machine scheduling problem in which the setup time and learning effect are considered, and the setup times are past-sequence-dependent. They showed that the problems to minimize the sum of quadratic job completion time, the total waiting time, the total absolute differences in waiting time, and the sum of earliness penalties subject to no tardy jobs, are polynomially solvable. Wang et al. [37] studied a single-machine problem with learning effect and discounted cost. They showed that the shortest processing time first (*SPT*) rule is the optimal policy for minimizing the discounted total completion time. They then illustrated an example to demonstrate that the discounted weighted shortest processing time first (*WDSPT*) rule is not the optimal policy for minimizing the discounted total weighted completion time. In addition, Mosheiov and Sidney [28] developed a learning model in which the learning effects are different depend on the jobs. They formulated the makespan scheduling problem with the job-dependent learning effects as an assignment problem and conducted a Hungarian method to solve the problem. And then Koulamas [18] proved that the problem proposed by Mosheiov and Sidney [28] can be solved in $O(n \log n)$ times under certain agreeable condition. Furthermore, Janiak and Rudek [14] proposed a new learning

effect model in which the rigorous constraints of the position-dependent approach are relaxed by assuming that each job creates a different experience for the processor. They also described the shape of the learning curve using a k -stepwise function. Hence, the diversified learning functions can be fitted by a mathematical model. Janiak and Rudek [15] proposed a new experience-based learning model where the job processing times are described by “S”-shaped functions and are dependent on the experience of the processor. They demonstrated that the makespan problem on a single processor is NP-hard or strongly NP-hard, and then provided a number of polynomially solvable cases. In addition, Huang et al. [13] investigated two resources constrained single-machine group scheduling problems in which the learning effect and deteriorating jobs are considered simultaneously. They proposed polynomial solutions under certain constraints to minimize the makespan and the resource consumption, respectively. Lee and Lai [20] considered both the effect of learning and deterioration in a scheduling model. The actual job processing time is a function on the processing times of scheduled jobs and its position in the schedule. They showed that some single-machine scheduling problems remain polynomial solvable. Toksari [33] addressed a single-machine scheduling problem with unequal release times for minimizing the makespan, in which the learning effect and the deteriorating jobs are concurrently considered. Several dominance criteria and the lower bounds are established to facilitate the branch-and-bound algorithm for deriving the optimal solution. Furthermore, Eren and Guner [9] studied a bi-criteria scheduling problem with a learning effect in an m -identical parallel machine environment, and the objective function is to minimize the weighted sum of the total completion time and total tardiness. They constructed a mathematical programming model to solve the problem. Toksari and Guner [34] considered a parallel machine earliness/tardiness scheduling problem involving different penalties under the effect of position-based learning and deterioration, and demonstrated that the optimal sequence is a V-shaped schedule under certain agreeable conditions.

As for the sum-of-processing-time-based learning model, Koulamas and Kyparisis [17] pointed out that employees learn more when executing jobs with a longer processing time. They introduced a sum-of-job-processing-time-based learning effect scheduling model and demonstrated that the makespan and the total completion time problems for the single machine and two-machine flowshop with ordered job processing times are polynomially solvable. Wu et al. [45] studied a total weighted completion time problem on a single machine with learning effect and ready times. A branch-and-bound algorithm was proposed to derive the optimal sequence, and the simulated annealing algorithm was implemented to obtain the near-optimal sequence. Furthermore, Cheng et al. [5] introduced a learning effect model on a single machine in which the actual job processing time is derived from the sum of the logarithm of the processing times of jobs already processed, and they show that the makespan and total completion time problems are polynomially solvable. Cheng et al. [4] proposed a two-agent scheduling problem with a truncated sum-of-processing-time-based learning effect on a single machine. A branch-and-bound algorithm was utilized to obtain the optimal solution for minimizing the total weighted completion time for the jobs of the first agent subject to no tardy job of the second agent. Wang [38] introduced an exponential sum-of-actual-processing-time-based learning effect into a single-machine scheduling problem. The special cases of the total weighted completion time problem and the maximum lateness problem are proved to be polynomial solvable under an adequate condition. Additionally, Wang et al. [41] demonstrated that, even with the effects of learning and deterioration on job processing times, the single-machine makespan problem remains polynomially solvable. Wang et al. [40] considered the weighted sum of completion times and the maximum lateness problem with the effect of learning and deterioration on a single machine where job processing times are defined as functions of their starting times and sequential positions.

In recent literature, the position-based and the sum-of-processing-time-based learning

have been discussed simultaneously. Yin et al. [49] examined some single-machine and m -machine flowshop problems with learning considerations where the learning effect is not only a function of the total normal processing times of jobs already processed, but also of the scheduled job position. Lee and Wu [22] presented a learning model that simultaneously combines the position-based learning and sum-of-processing-time-based learning models. They then demonstrated that the single-machine makespan and the total completion time problems are polynomially solvable, and provided polynomial-time optimal sequences for minimizing the makespan and total completion time under certain conditions in a flowshop environment. Furthermore, Wang and Li [35] studied a single machine scheduling problem with past-sequence dependent setup times in which the position-based and time-dependent learning effects are simultaneously considered. They proved that the makespan, total completion time and total lateness problems can be solved by the smallest processing time first (*SPT*) rule. Lai and Lee [19] addressed a general scheduling model in which the position-based and the sum-of-processing-time-based learning effects are concurrently considered. They showed that most of the models in the literatures are special cases of the model they proposed.

The concept of learning effect in a flowshop environment has been relatively neglected. However, Wu et al. [47] studied the maximum tardiness problem with the position-based learning effect in a two-machine flowshop environment. They implemented a branch-and-bound algorithm to obtain the optimal sequence, and a simulated annealing algorithm to obtain the near-optimal sequence. Li et al. [25] discussed a two-machine flowshop scheduling problem with a truncated learning effect which considers the position of the job in a schedule and the control parameter. Then the branch-and-bound and three simulated annealing algorithms were conducted to seek the optimal and near-optimal solutions. In addition, Lee and Wu [21] considered a two-machine flowshop problem with learning effect for minimizing the total completion time. They utilized two lower bounds and

several dominance properties to construct a branch-and-bound algorithm to obtain the optimal sequence, and established a heuristic algorithm to obtain the near-optimal sequence. Chen et al. [3] considered a bi-criteria two-machine flowshop scheduling problem with the position-based learning effect when the goal is to minimize both the total completion time and the maximum tardiness. They proposed a branch-and-bound algorithm and two heuristic algorithms to obtain the optimal and near-optimal sequences. Furthermore, Wang and Xia [42] studied flowshop problems with learning effect. They gave the worst-case bound of the shortest processing time first (*SPT*) algorithm for the makespan and the total flow time problems, then illustrated examples to show that the Johnson's rule is not optimal for the makespan problem in a two-machine environment with learning consideration. Eventually, they demonstrated that two special cases remained polynomially solvable for the makespan and total completion time problems. Additionally, Wu and Lee [46] investigated a flowshop problem with learning considerations to minimize the total completion time. They implemented a branch-and-bound algorithm and heuristic algorithms to seek the optimal and near-optimal sequences, respectively.

Because of obtaining optimal sequences in scheduling problems within a flowshop environment is usually complicated, numerous works have focused on identifying efficient near-optimal sequences. Nawaz et al. [27] considered an m -machine flowshop problem for minimizing the makespan, and claimed that jobs with larger total normal processing time should be prioritized over jobs with smaller total normal processing times. They demonstrated that their proposed algorithm performs particularly well on large job-sized problems. Afterward, Liu and Ong [26] and Ruiz and Maroto [31] claimed that the algorithm developed by Nawaz et al. [27] is superior to other existing polynomial algorithms for the m -machine flowshop makespan problem. Furthermore, Rajendran and Ziegler [30] developed an algorithm for solving the weighted total completion time minimization problem in an m -machines flowshop environment. Their algorithm first generates m sequences by assigning

different weights to each machine. The sequence with the minimal total weighted completion time is then selected as the seed sequence, and an improvement scheme is employed. Woo and Yim [43] provided an algorithm for minimizing the mean flow time in an m -machine flowshop environment. Their algorithm selects a job among excluded jobs for insertion into the current partial sequence. Whenever a new partial schedule is constructed, their algorithm assesses all the possible sequences by inserting an unscheduled job into one of all slots in the current sequence at a time. The partial sequence with the least mean flow time is selected. In addition, Framinan and Leisten [11] considered an m -machine flowshop problem to minimize the mean flow time. They proposed an efficient constructive heuristic algorithm based on the concept of the algorithm of Nawaz et al. [27]. They further performed a general pairwise interchange movement to boost the quality of the partial sequences in all the iterations. Framinan et al. [10] presented a review and classification for the heuristic algorithms with a makespan objective. They distinguished a given constructive heuristic algorithm into three phases, which are index development, solution construction and solution improvement. Furthermore, Wang et al. [36] proposed a modified global-best harmony search algorithm to obtain the near-optimal solution for dealing with a makespan scheduling problem in a blocking permutation flowshop environment. The algorithm they proposed was demonstrated to outperform certain existing meta-heuristics. Zhang and Li [50] addressed an estimation of distribution algorithm for a flowshop scheduling problem with the objective of minimizing the total flowtime. They showed that the proposed algorithm could improve some current best solutions for Taillard benchmark instances.

1.3 Research objectives and methodologies

Two m -machine flowshop scheduling problems are proposed in this dissertation in which the models are based on Biskup [1]. The model proposed in Biskup [1] is presented as

$p_{j,r} = p_j r^a$ where $p_{j,r}$ denotes the actual processing time of job j at r th scheduled position, p_j denotes the normal processing time of job j , and $a \leq 0$ denotes the learning index. The decreasing level for the curve of r^a descends as r increases, and it conforms to the phenomenon that the improvement of the worker's skill is unobvious after the worker is proficient at the jobs. Therefore, the learning model proposed in Biskup [1] is reasonable and regarded as a theoretical learning model in many studies. In addition, the production environment for the model proposed in Biskup [1] could be regarded as the handicraft because of the learning effect is occurred in whole process when dealing with a job. Meanwhile, the learning model proposed in Biskup [1] might be considered as the reduced learning model for the industrial manufacturing.

In this dissertation, the types of learning effect of the two problems belong to position-based learning. Furthermore, the learning effects are identical on all machines in the first problem, and are varied on different machines in the second problem. Since the makespan and the total completion time are the objective functions that are widely used performance measures in the scheduling literature, the objective in this dissertation of the first problem is to minimize the makespan, and of the second problem is to minimize the bi-criteria function which is modeled as the weighted sum of the total completion time and the makespan.

While the number of the machines is more or equal than three, Garey et al. [12] demonstrated that the flowshop scheduling problem for minimizing the makespan without the learning effect is an NP-hard problem. In addition, the total completion time minimization problem is proved to be an NP-hard problem without considering the learning effect when the number of the machines is more or equal than two (Lenstra et al. [24]). Therefore, the makespan and the bi-criteria minimization problems in this dissertation are both NP-hard problems. Then the branch-and-bound algorithm is a feasible approach for deriving the

optimal sequence. In the literature with respect to the flowshop scheduling problems without learning effect, Chung et al. [7] studied an m -machine flowshop scheduling problem to minimize the total completion time. They proposed a branch-and-bound algorithm that incorporates a dominance property and an innovative lower bound to seek the optimal sequence. Thereafter, Chung et al. [8] modified the efficient property in Chung et al. [7] to deal with the flowshop scheduling problem for minimizing the total tardiness. Therefore, in this dissertation, a branch-and-bound algorithm is conducted to obtain the optimal sequence, in which the dominance properties are established based on the concept of Chung et al. [7].

Seeking for the optimal sequence of scheduling problems generally requires considerable computational time and memory for larger job-sized problems. Thus this dissertation also focuses on assessing the performances of efficiency when applying economical heuristic algorithms with the learning effect to solve the proposed problem. And then two well-known heuristic algorithms proposed from Nawaz et al. [27] and Framinan and Leisten [11] are adapted for obtaining the near-optimal sequence. Additionally, two meta-heuristic algorithms are also utilized to yield the near-optimal solutions which are simulated annealing and genetic algorithms. Eventually, the accuracy and the comparison for the priorities among proposed heuristic and meta-heuristic algorithms are discussed in this dissertation.

Chapter 2

Algorithms

2.1 Branch-and-bound algorithms

In this dissertation, two NP-hard problems are studied. In order to seek the optimal sequence, we conduct a branch-and-bound algorithm incorporated with a dominance property and a lower bound. In branch-and-bound algorithm, a given node indicates a sequence with scheduled jobs, and the nodes can be eliminated by verifying the dominance property or evaluating the lower bound. The dominance property is utilized to prove that the given node is dominated by another node. Furthermore, the lower bound is the underestimated value of the objective function based on the given node. Therefore, when the given node is dominated or its lower bound is larger than a known objective value, the given node and its offspring are eliminated in the branching tree. In addition, the branching procedure proposed in this dissertation adopts the depth-first search, and assigns jobs in a forward manner starting from the first position. The advantages of the depth-first search are less number of dynamic nodes and seeking the bottom node rapider to derive the feasible sequence. The detailed procedure of the proposed branch-and-bound algorithm is described as follows.

- Step 1:** Generate a near-optimal sequence and solution as the initial incumbent sequence and solution by implementing the heuristic and meta-heuristic algorithms.
- Step 2:** Expand the branching tree from node $(-, -, \dots, -)$ to node $(1, -, \dots, -)$, then to node $(1, 2, -, \dots, -)$, and finally to node $(n, n-1, \dots, 1)$.
- Step 3:** If the current node is a complete sequence, go to Step 6. Otherwise, go to Step 4.
- Step 4:** Apply the dominance property to identify the current node. If it is a dominated node, eliminate the node and its offspring in the branching tree, then go to Step 2. Otherwise, go to Step 5.

Step 5: Evaluate the lower bound of the objective value for the current node. If the lower bound for the current node is larger than the incumbent solution, eliminate the node and its offspring in the branching tree, then go to Step 2. Otherwise, go to Step 7.

Step 6: If the objective value of the complete sequence is smaller than the incumbent solution, replace the incumbent sequence and solution with the sequence and solution of the current node. Otherwise, eliminate it.

Step 7: If there is no more node can be expanded, the final incumbent sequence is set as the optimal sequence. Otherwise, go to Step 2.

Eventually, a flowchart is drawn in Fig 2.1. to illustrate the detailed procedure of the proposed branch-and-bound algorithm.



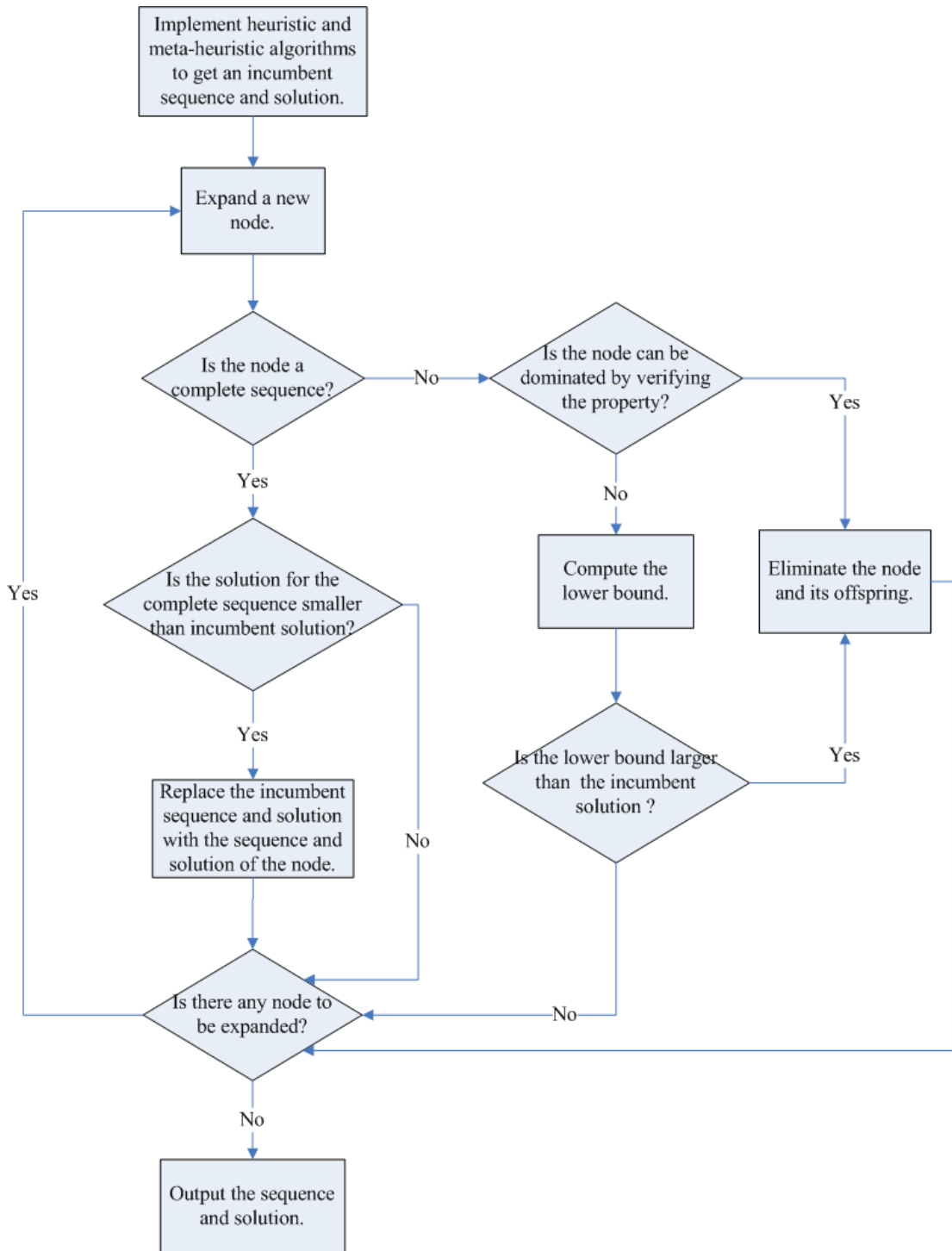


Fig 2.1. The flowchart of the proposed branch-and-bound algorithm

2.2 Heuristic algorithms

While the number of jobs increases, obtaining the optimal solution of an NP-hard scheduling problem is time-consuming. Therefore, many studies are devoted to develop efficient heuristic algorithms to derive the near-optimal solution. In addition, the objective functions in this dissertation consist of the makespan and the total completion time. Therefore, *NEH* and *FL* denote the heuristic algorithms which is respectively adapted from the heuristic algorithm proposed in Nawaz et al. [27] and Framinan and Leisten [11], by considering the learning effect and adjusting the objective function. Eventually, the procedures of *NEH* and *FL* are detailed as follows.

NEH algorithm:

Step 1: Set sequence *PS* and *US* with empty set.

Step 2: Arranging the jobs in descending order of the total normal processing times (i.e.

$$\sum_{i=1}^m p_{i,j} \text{ for } j = 1, 2, \dots, n. \text{ See subsection 3.1), and schedule the jobs into } US.$$

Step 3: Set $k = 1$.

Step 4: Select the first job from *US* into *PS*, and remove the job from *US*.

Step 5: If $k = 1$, go to Step 4. Otherwise, generate k sequence by respectively inserting the job into each slot of *PS*.

Step 6: Select the sequence with the least objective value among k candidate sequences and update the sequence as *PS*.

Step 7: Set $k = k + 1$. If $k \leq n$, go to Step 4. Otherwise, the near-optimal sequence is set as *PS*.

FL algorithm:

Step 1: Set sequence PS and US with empty set.

Step 2: Arranging the jobs in ascending order of the total normal processing times, and schedule the jobs into US .

Step 3: Set $k = 1$.

Step 4: Select the first job from US into PS , and remove the job from US .

Step 5: If $k = 1$, go to Step 4. Otherwise, generate k sequence by respectively inserting the job into each slot of PS .

Step 6: Select the sequence with the least objective value among k candidate sequences and update the sequence as PS .

Step 7: If $k < 3$, go to Step 8. Otherwise, generate $\frac{k(k-1)}{2}$ sequences based on PS by performing pairwise interchange procedure. Then select the sequence with the least objective value and set as sequence PS' . If PS can be dominated by PS' in terms of the objective value, replace PS with PS' .

Step 8: Set $k = k + 1$. If $k \leq n$, go to Step 4. Otherwise, the near-optimal sequence is set as PS .

2.3 Meta-heuristic algorithms

Pinedo [29] stated that the heuristic algorithms of the constructive type start without a sequence and gradually construct a sequence by adding one job at a time. They usually can obtain a solution in a moment. However, the quality of the solutions obtained by the heuristic algorithms of the constructive type is improvable, especially when the priority of the jobs is difficult to estimate. The reason is that the solutions space for the heuristic algorithms of the constructive type is relatively narrow. Therefore, two meta-heuristic algorithms are implemented to obtain the near-optimal solutions because of their larger solutions spaces, which are simulated annealing and genetic algorithms. The procedure of seeking the solution by meta-heuristic algorithms is iteratively trying to improve a candidate solution in terms of a given measure of quality. The advantages of the meta-heuristic algorithms are few or no assumptions in searching process, and a larger space of candidate solutions. Eventually, two meta-heuristic algorithms proposed in this dissertation are detailed below.

Simulated Annealing (SA)

A description of the procedure in proposed simulated annealing is presented as follows. An incumbent sequence is generated at first, and then a new sequence is created based on the incumbent one by the neighborhood generation. The incumbent sequence is replaced with the new sequence when each one of two conditions occurred, that is (1) the objective value of the new sequence is smaller than that of the incumbent sequence, and (2) the acceptance probability is larger than a given value. Eventually, the process of proposed simulated annealing is stopped by the terminating condition, and then the final incumbent sequence is set as the near-optimal sequence. The elaboration of the simulated annealing proposed in this dissertation includes:

- (1) **Incumbent sequence:** The incumbent sequence is generated randomly.
- (2) **Neighborhood generation:** Two jobs of incumbent sequence are randomly selected and exchanged to yield the new sequence.
- (3) **Acceptance probability:** The probability of acceptance is yielded from an exponential distribution, that is $P(\text{accept}) = \exp(-\alpha \times \Delta)$, where α denotes the control parameter and Δ denotes the increment of the objective value from the incumbent to the new sequence. Furthermore, the control parameter is set as $\frac{k}{\beta}$, where k is the number of cumulated iterations to date and β is an experimental factor. And then we chose $\beta = 65000$ after some pretests. If the new sequence is larger than the incumbent one, the new sequence is accepted when $P(\text{accept}) > r$, where r is a uniform random number between 0 and 1.
- (4) **Terminating condition:** The seeking process is terminated after $500n$ iterations because of the preliminary tests reveal that the objective value is steady after $500n$ iterations, where n is the number of jobs.

The advantage of SA is to avoid getting trapped in a local optimum. The value of β is initially set to a high level so that a neighborhood exchange happens frequently in early iterations, and the acceptance probability is gradually lowered when the k increases so that it becomes more difficult to exchange in later iterations unless a better sequence is yielded.

Genetic algorithm (GA)

In this dissertation, a genetic algorithm is utilized to yield the near-optimal solution. The basic idea of the genetic algorithm is to generate a population with some chromosomes as the parents, then implement the crossover and mutation operations to produce a new population as the offspring, and choose the chromosome with the best performance with regard to the objective value after some generations. The segments of proposed genetic algorithm are listed as follows.

- (1) **Encoding:** The encoding method is to generate n uniform random numbers from $(0, 1)$ as the genes in a chromosome, in which the job order is set as the non-decreasing order of the genes. For example, the chromosome $(0.23, 0.78, 0.32, 0.14)$ denotes a job sequence of $(4, 1, 3, 2)$ with four jobs.
- (2) **Population size:** The population size indicates the number of the chromosomes in a generation. For a large population size, it is easier to obtain a better solution, but time-consuming. The population size N is set as 150 after some preliminary tests.
- (3) **Fitness value:** The fitness values are evaluated to indicate the probabilities of selecting the chromosomes. Since the problem is to minimize the objective value, the fitness value of a given chromosome should be a decreasing function of its objective value. Therefore, the fitness value of the chromosome is calculated as $f_k / \sum_{j=1}^N f_j$ for $k = 1, 2, \dots, N$, where the f_k denotes the reciprocal of the objective value of the k th chromosome.
- (4) **Selection:** A roulette wheel method is used in which the chromosomes with larger fitness values have larger areas in the roulette wheel and have higher chances to be selected. The selection process is executed by spinning the roulette wheel, and only a chromosome is selected in each spin.
- (5) **Crossover:** Crossover operation is to produce the chromosomes of offspring from the chromosomes of parents. In this dissertation, two chromosomes in parents are selected to

generate two new chromosomes in offspring by utilizing one-point crossover, in which a cut point are randomly selected and the parts of these two chromosomes in parents are exchanged to generate new chromosomes. For example, two chromosomes in parents are presented as (0.53, 0.26, 0.72, 0.44, 0.69) and (0.91, 0.08, 0.37, 0.29, 0.55), and the new chromosomes are (0.53, 0.26, 0.37, 0.29, 0.55) and (0.91, 0.08, 0.72, 0.44, 0.69) if the randomly selected cut point is the second position. Furthermore, the crossover rate is chosen as 0.85 in this dissertation after some pretests.

- (6) **Mutation:** Mutation operation is used to prevent getting trip in a local optimum. In this dissertation, the procedure of mutation is to randomly select a gene in a given chromosome, and replace it with a random uniform number from 0 to 1. The mutation rate is set as 0.3 in this dissertation to determine whether the chromosome is mutated.
- (7) **Evolution:** In order to maintain the superiority of the chromosomes, a part of chromosomes with larger fitness values in parents are retained in the next generation. Meanwhile, the chromosomes with smaller fitness values in the offspring are eliminated as a consequence. Furthermore, the evolution rate is set as 0.5, which means 50% of the chromosomes in the parents are retained to the next generation.
- (8) **Termination:** The proposed genetic algorithm is terminated after 50 generations after some pretests.

Eventually, the flowchart of proposed genetic algorithm is shown in Fig 2.2.

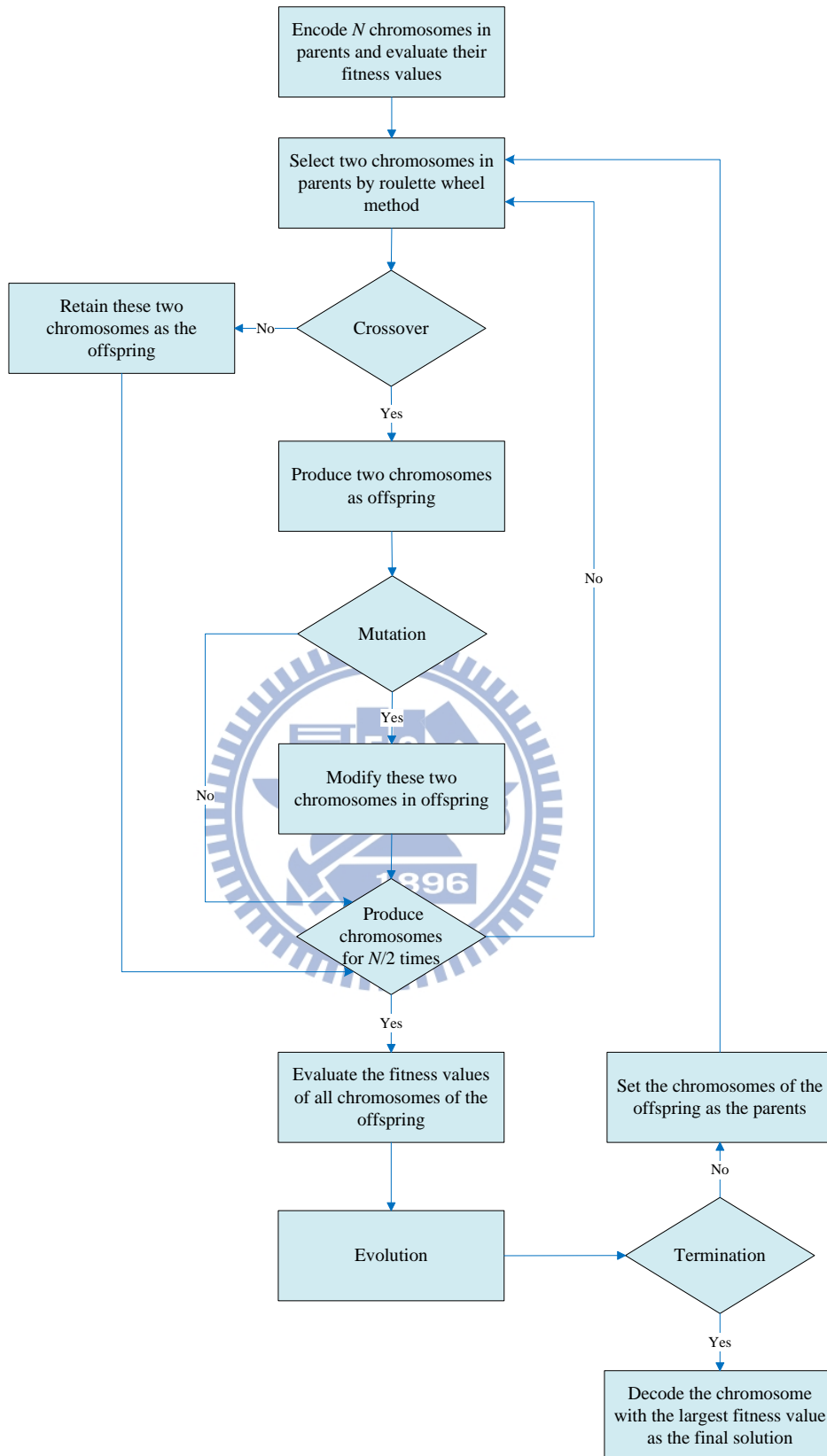


Fig 2.2. The flowchart of the proposed genetic algorithm

Chapter 3

Makespan minimization for m -machine flowshop scheduling problem with position-based learning effects

3.1 Notations and problem statement

The notations used throughout this chapter are summarized as follows.

n : Number of jobs.

m : Number of machines.

N : Set of jobs, i.e., $N = \{1, 2, \dots, n\}$.

M_i : i th machine, $i = 1, 2, \dots, m$.

$p_{i,j}$: Normal processing time of job j on M_i .

$p_{i,j,r}$: Actual processing time of job j on M_i if placed at r th position in a sequence.

a : Learning index with $a < 0$.

S : Subset of N with s scheduled jobs.

U : Subset of N with $n - s$ unscheduled jobs.

σ : A partial sequence of set S .

[]: The symbol which signify the order of jobs in a sequence.

$C_{i,[r]}(\sigma)$: Completion time of the job scheduled in the r th position on M_i in sequence

σ .

$G_j(u, v)$: Total normal processing time of job j from M_u to M_v , where $u \leq v$, i.e.,

$$G_j(u, v) = \sum_{l=u}^v p_{l,j}.$$

$B_{i,[r]}$: Earliest starting time at r th position on M_i .

$F_{i,[r]}$: Earliest completion time at r th position on M_i .

LB : The lower bound for a given node.

The problem formulation of the m -machine flowshop environment with learning effects is described as follows. Suppose that there are n jobs in set N , to be processed on m machines. Each job j comprises m operations $O_{1,j}, O_{2,j}, \dots, O_{m,j}$, where $O_{i,j}$ has to be processed on M_i for $i=1,2,\dots,m$ and $j=1,2,\dots,n$. Processing of operation $O_{i+1,j}$ must start only after the completion of $O_{i,j}$. Furthermore, the flowshop environment considers a schedule in which the job sequence is identical on all the machines. The actual processing time $p_{i,j,r}$ of job j on M_i is a function that depends on its position r in the sequence, i.e.,

$$p_{i,j,r} = p_{i,j}r^a,$$

where $i=1,2,\dots,m$, $j,r=1,2,\dots,n$.

This chapter attempts to identify a sequence for minimizing the makespan. Given n jobs in Set N , and τ denotes one complete sequence of all permutations. The objective of this chapter is to derive a sequence τ^* such that $C_{m[n]}(\tau^*) \leq C_{m[n]}(\tau)$ for any sequence τ .

3.2 Dominance property

The following theorem provides a criterion for discriminating dominance relationships between two different sequences which are made up of the same job set.

Theorem 3.1: Let σ_1 and σ_2 denote two partial sequences with s jobs of set S . If

$$\max_{1 \leq i \leq m} \{C_{i,[s]}(\sigma_1) - C_{i,[s]}(\sigma_2)\} < 0, \text{ then } \sigma_1 \text{ dominates } \sigma_2.$$

Proof: Let π denote a partial sequence with $n-s$ jobs of set U , and sequence π is scheduled immediately behind sequence σ_1 and σ_2 into the sequence $S_1 = (\sigma_1, \pi)$ and

$S_2 = (\sigma_2, \pi)$, respectively. Then for $1 \leq u \leq m$, we have the completion time of the job scheduled in the n th position on M_u in S_1 and is

$$\begin{aligned} C_{u,[n]}(S_1) &= \max_{1 \leq v \leq u} \{C_{v,[n-1]}(S_1) + G_{[n]}(v, u) \times n^a\} \\ &= C_{v_1,[n-1]}(S_1) + G_{[n]}(v_1, u) \times n^a \text{ for some } v_1 \text{ where } 1 \leq v_1 \leq u. \end{aligned}$$

Similarly, the completion time of the job scheduled in the n th position on M_u in S_2 is

$$\begin{aligned} C_{u,[n]}(S_2) &= \max_{1 \leq v \leq u} \{C_{v,[n-1]}(S_2) + G_{[n]}(v, u) \times n^a\} \\ &= C_{v_2,[n-1]}(S_2) + G_{[n]}(v_2, u) \times n^a \text{ for some } v_2 \text{ where } 1 \leq v_2 \leq u. \end{aligned}$$

Then we have

$$C_{u,[n]}(S_2) \geq C_{v_1,[n-1]}(S_2) + G_{[n]}(v_1, u) \times n^a \text{ for } v_1 \neq v_2.$$

Therefore, we have

$$\begin{aligned} C_{u,[n]}(S_1) - C_{u,[n]}(S_2) &\leq [C_{v_1,[n-1]}(S_1) + G_{[n]}(v_1, u) \times n^a] - [C_{v_1,[n-1]}(S_2) + G_{[n]}(v_1, u) \times n^a] \\ &\leq \max_{1 \leq i \leq m} \{C_{i,[n-1]}(S_1) - C_{i,[n-1]}(S_2)\}. \end{aligned}$$

An induction argument is conducted. Then we have

$$C_{u,[n]}(S_1) - C_{u,[n]}(S_2) \leq \max_{1 \leq i \leq m} \{C_{i,[s]}(S_1) - C_{i,[s]}(S_2)\}.$$

If $\max_{1 \leq i \leq m} \{C_{i,[s]}(S_1) - C_{i,[s]}(S_2)\} < 0$, then S_1 dominates S_2 .

The proof is completed.

In order to apply the above theorem in the proposed branch-and-bound algorithm, the following property requires considering two consecutive jobs, as presented below.

Property 3.1: Let J_x and J_y denote two jobs of set S , and σ_{s-2} denote a sequence with

$s-2$ jobs excluding J_x and J_y of set S . If $\max_{1 \leq i \leq m} \{C_{i,[s]}(\sigma_{s-2}, J_x, J_y) - C_{i,[s]}(\sigma_{s-2}, J_y, J_x)\} < 0$

, then sequence (σ_{s-2}, J_x, J_y) dominates (σ_{s-2}, J_y, J_x) .

3.3 Lower bound

For a given node in the branch-and-bound algorithm, the lower bound is designed to underestimate the objective function by utilizing the information of its unscheduled jobs, and the lower bound is less than or equal to the objective value of the optimal sequence based on the node. Consequently, when the lower bound of a given node is larger than the objective value of a known sequence, the optimal sequence based on the node is dominated by the known sequence, and the given node and its offspring are not the candidates for the optimal sequence.

In this subsection, we propose a lower bound for eliminating nodes in the branching tree, and the lower bound is evaluated using the concept developed by Chung et al. [7]. The lower bound for Chung et al. [7] is a machine-based lower bound. The main idea of their lower bound is assuming that the given machine has unit capacity and the machines behind it have infinite capacity. Hence, the procedure in Chung et al. [7] for estimating the marginal lower bound based on the given machine is to compute the earliest starting times for all remaining positions on the machine at first, and to sum up these starting times, and all the processing times of the machine, and that behind the machine for unscheduled jobs. Finally, the lower bound is determined as the maximal marginal lower bound. Instead of the total completion time, we adapt the procedure in Chung et al. [7] which estimates the earliest starting time with learning effect, when the objective is to minimize the makespan. The proposed lower bound is summarized as follows. Let $p_{i,(j)}$ represent the normal processing times on M_i , which are based on non-descending order of all $p_{i,j}$ from set U for $j = 1, 2, \dots, n-s$. i.e., $p_{i,(1)} \leq p_{i,(2)} \leq \dots \leq p_{i,(n-s)}$, where $i = 1, 2, \dots, m$. $G_{(1)}(u, v)$ denotes the smallest total normal processing time between M_u and M_v from set U . Let $E_{i,[s+1]}$ denote the actual starting time of $s+1$ th job on M_i . By definition, we have

$$E_{1,[s+1]} = C_{1,[s]}(\sigma)$$

and

$$E_{i,[s+1]} = \max \left\{ \max_{1 \leq u \leq i-1} \left\{ E_{u,[s+1]} + G_{[s+1]}(u, i-1) \times (s+1)^a \right\}, C_{i,[s]}(\sigma) \right\}, \text{ where } i = 2, 3, \dots, m.$$

For the first machine, the earliest starting time is the same as the actual starting time of $s+1$ th job (i.e. $B_{1,[s+1]} = E_{1,[s+1]}$). Then

$$\begin{aligned} E_{2,[s+1]} &= \max \left\{ B_{1,[s+1]} + p_{1,[s+1]} \times (s+1)^a, C_{2,[s]}(\sigma) \right\} \\ &\leq \max \left\{ B_{1,[s+1]} + p_{1,(1)} \times (s+1)^a, C_{2,[s]}(\sigma) \right\}. \end{aligned}$$

Therefore, $B_{2,[s+1]}$ is evaluated as $\max \left\{ B_{1,[s+1]} + p_{1,(1)} \times (s+1)^a, C_{2,[s]}(\sigma) \right\}$. By induction, we have

$$B_{i,[s+1]} = \max \left\{ \max_{1 \leq u \leq i-1} \left\{ B_{u,[s+1]} + G_{(1)}(u, i-1) \times (s+1)^a \right\}, C_{i,[s]}(\sigma) \right\} \text{ for } i = 2, 3, \dots, m.$$

Since the learning effect is considered, we have $F_{i,[s+j]} = B_{i,[s+1]} + \sum_{l=1}^j p_{i,(l)} (s+l)^a$. For the first machine, the earliest starting time of n th job is the earliest completion time of $(n-1)$ th job (i.e. $B_{1,[n]} = F_{1,[n-1]}$). In the context of Chung et al. [7] for unscheduled jobs, besides $(s+1)$ th job on the second to the final machine, the procedure of computing the earliest starting time only considers the earliest completion time on the current machine, and that immediately ahead of the machine (i.e. $E_{i,[s+j]} = \max \left\{ F_{i,[s+j-1]}, F_{i-1,[s+j]} \right\}$). However, it may have the contradiction that the earliest starting time on the current machine is smaller than that on the preceding machines for the third and late machine. Therefore, to overcome the contradiction, we have

$$B_{i,[n]} = \begin{cases} \max \left\{ F_{i,[n-1]}, F_{i-1,[n]} \right\} & , \text{ where } i = 2 \\ \max \left\{ F_{i,[n-1]}, F_{i-1,[n]}, B_{i-1,[n]} + p_{i-1,(1)} \times n^a \right\} & , \text{ where } i = 3, 4, \dots, m \end{cases}$$

Then the marginal lower bound is evaluated as $B_{i,[n]} + G_{(1)}(i, m) \times n^a$. Eventually, the lower

bound in this chapter is represented as $\max\left\{\max_{1 \leq i \leq m}\{B_{i,[n]} + G_{(1)}(i, m) \times n^a\}, F_{m,[n]}\right\}$, and the detailed procedure for estimating the lower bound is presented as follows.

Step 1: Set $i = 1$, $B_{1,[s+1]} = C_{1,[s]}(\sigma)$, and go to Step 3.

Step 2: Compute $B_{i,[s+1]} = \max\left\{\max_{1 \leq u \leq i-1}\{B_{u,[s+1]} + G_{(1)}(u, i-1) \times (s+1)^a\}, C_{i,[s]}(\sigma)\right\}$

Step 3: Compute $F_{i,[s+j]} = B_{i,[s+1]} + \sum_{l=1}^j p_{i,(l)}(s+l)^a$ for $j = n-s-1$ and $n-s$.

Step 4: If $i = 1$, set $B_{1,[n]} = F_{1,[n-1]}$ and go to Step 6. Otherwise, go to Step 5.

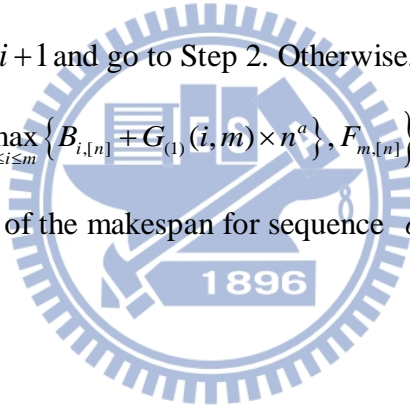
Step 5: If $i = 2$, set $B_{i,[n]} = \max\{F_{i,[n-1]}, F_{i-1,[n]}\}$. Otherwise, set

$$B_{i,[n]} = \max\{F_{i,[n-1]}, F_{i-1,[n]}, B_{i-1,[n]} + p_{i-1,(1)} \times n^a\}.$$

Step 6: If $i < m$, set $i = i + 1$ and go to Step 2. Otherwise, go to Step 7.

Step 7: Set $LB = \max\left\{\max_{1 \leq i \leq m}\{B_{i,[n]} + G_{(1)}(i, m) \times n^a\}, F_{m,[n]}\right\}$.

Step 8: The lower bound of the makespan for sequence σ is obtained as LB .



3.4 Computational results

In this section, several computational experiments are conducted to assess the performance of the branch-and-bound, the heuristic and meta-heuristic algorithms proposed in this chapter. All the algorithms are coded in Fortran 90 and run on a personal computer with 2.89 GHz AMD Athlon™ II X4 635 Processor and 3.25GB RAM with Windows XP. The normal processing time of all operations are randomly generated from a discrete uniform distribution over 1 to 100. First of all, the influence of the learning effect is examined in Table 3.1, in which the number of jobs is fixed at 10, three different levels of the learning effect are set as 90%, 80% and 70% (which corresponds to $a = -0.152$, $a = -0.322$, and $a = -0.515$.), and 100 replications are randomly generated of each experimental condition. Therefore, a total of 600 instances are tested and the mean optimal makespans are recorded in Table 3.1. Furthermore, the optimal sequence derived from the proposed problem without learning effect under each instance, is used to calculate the makespan of the proposed problem and the mean makespans and the mean and maximum error percentages are listed in Table 3.1. For each instance, the error percentage of is calculated as $\frac{O - O^*}{O^*} \times 100\%$, where O denotes the value of the makespan calculated by the sequence derived without the learning effect and O^* denotes the optimal makespan. As shown in Table 3.1, it reveals that the influence of the learning effect is notable with regard to the mean error percentages. Additionally, the influence of the learning effect is higher with the stronger learning effect.

Table 3.1. The influence of the learning effect on optimal solution ($n = 10$)

m	$a(\%)$	mean optimal makespan	Use the optimal sequence which is derived without the learning effect		
			mean makespan	Error percentage	
				mean	max
3	90%	487.1	563.4	15.799	31.804
	80%	375.1	531.0	42.506	75.273
	70%	285.0	528.5	86.911	144.845
5	90%	580.5	634.5	9.447	26.843
	80%	466.3	583.7	25.705	62.889
	70%	376.8	541.3	44.478	107.165

In order to test the efficiency of the proposed property and the lower bound, a computational experiment is implemented with fixed job size at 10, two different machine sizes at 3 and 5, 100 replications, and three levels of the learning effect at 90%, 80% and 70%. The results are listed in Table 3.2, in which B_P denotes the branch-and-bound algorithm with only the property, B_L denotes the branch-and-bound algorithm with only the lower bound, and B_P+L denotes the branch-and-bound algorithm with both the property and the lower bound. In addition, the mean number of nodes and the mean execution time are recorded. Meanwhile, the mean execution time for the enumeration method is also recorded. As shown in Table 3.2, the efficiency of the property and the lower bound in the branch-and-bound algorithm are significant in terms of the mean execution time by comparison with the enumeration method. Furthermore, the lower bound is more effective than the property in terms of the mean number of nodes and the mean execution time, and the phenomenon is notable when the learning effect is stronger. However, the most efficient performance is exhibited when B_P+L is implemented in terms of the mean number of nodes and the mean execution time. Therefore, the branch-and-bound algorithm with both the property and the lower bound is recommended for the succeeding computational experiment in this chapter.

Table 3.2. The performance of the property and the lower bound for the branch-and-bound algorithm
($n = 10$)

m	$a(\%)$	Number of mean nodes			Mean CPU times			
		B_P	B_L	B_P+L	B_P	B_L	B_P+L	<i>Enumeration</i>
3	90%	257236.9	917.7	450.4	4.234	0.031	0.017	15.504
	80%	183932.9	162.6	129.6	3.083	0.007	0.006	15.421
	70%	111829.0	92.7	78.2	1.949	0.005	0.004	15.379

5	90%	368537.7	945.1	771.2	10.067	0.067	0.056	25.148
	80%	250310.5	350.0	310.5	6.892	0.027	0.027	25.051
	70%	146816.5	134.3	122.3	4.031	0.012	0.012	24.806

We use four job sizes ($n = 12, 14, 16$ and 18) and two different machine sizes ($m = 3$ and 5) to yield the optimal solution and test the accuracy of all the proposed heuristic and meta-heuristic algorithms. Furthermore, to examine the influence of the learning effect, the three levels of the learning effect are taken to be 90%, 80%, and 70%. Consequently, 24 experimental conditions are examined, and 100 replications are randomly generated for each condition. A total of 2,400 instances are generated and the results are listed in Table 3.3. The mean and the standard deviation of the number of nodes and of the execution time for the proposed branch-and-bound algorithm are recorded. In addition, the mean and standard deviation of the error percentages for the heuristic and meta-heuristic algorithms are also recorded. For each instance, the error percentage of the given heuristic algorithm is calculated as

$$\frac{V - V^*}{V^*} \times 100\%,$$

where V denotes the value of the makespan generated by the heuristic or meta-heuristic algorithm and V^* denotes the optimal makespan obtained by the branch-and-bound algorithm.

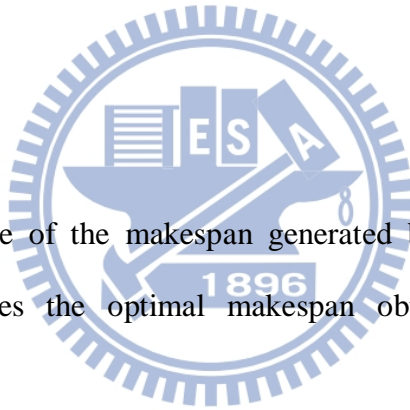


Table 3.3. The performance of branch-and-bound algorithm and heuristic algorithms of different parameters

<i>n</i>	<i>m</i>	<i>a</i> (%)	Branch-and-bound algorithm								Heuristic algorithms							
			Number of nodes					CPU times			Error percentages (%)							
			Mean	s.d.	Q1	Q2	Q3	Number of outliers	Mean	s.d.	<i>NEH</i>		<i>FL</i>		<i>SA</i>		<i>GA</i>	
											Mean	s.d.	Mean	s.d.	Mean	s.d.	Mean	s.d.
12	3	90%	56719.8	384694.2	78	288	1083	20	2.20	14.30	0.0134	0.0151	0.0062	0.0101	0.0055	0.0101	0.0061	0.0067
		80%	1192.9	3480.4	86	299	612	15	0.07	0.17	0.0193	0.0139	0.0064	0.0115	0.0094	0.0093	0.0071	0.0077
		70%	521.4	849.3	71	255	550	11	0.03	0.05	0.0359	0.0246	0.0069	0.0089	0.0130	0.0164	0.0103	0.0059
	5	90%	9448.4	34991.5	379	1093	4818	12	0.88	3.17	0.0247	0.0206	0.0146	0.0156	0.0132	0.0137	0.0109	0.0104
		80%	2772.7	9635.3	158	510	1722	12	0.28	0.84	0.0305	0.0202	0.0133	0.0143	0.0162	0.0135	0.0064	0.0095
		70%	583.2	1109.3	64	188	630	11	0.06	0.10	0.0426	0.0308	0.0119	0.0118	0.0212	0.0205	0.0070	0.0079
14	3	90%	167322.1	854106.4	200	1442	5900	14	8.63	44.15	0.0135	0.0116	0.0066	0.0098	0.0055	0.0077	0.0042	0.0065
		80%	19161.7	124696.6	295	988	5848	6	1.08	6.40	0.0241	0.0172	0.0068	0.0079	0.0130	0.0115	0.0063	0.0053
		70%	1720.3	2910.8	131	509	1623	12	0.15	0.23	0.0411	0.0244	0.0080	0.0075	0.0157	0.0163	0.0077	0.0050
	5	90%	301967.5	1838143.9	1583	4289	15460	20	27.97	151.64	0.0253	0.0195	0.0146	0.0129	0.0114	0.0130	0.0099	0.0086
		80%	9213.4	22874.2	622	2016	5053	19	1.26	2.93	0.0336	0.0196	0.0152	0.0144	0.0202	0.0131	0.0081	0.0096
		70%	6369.0	33345.7	486	1370	3463	11	0.87	3.80	0.0513	0.0249	0.0105	0.0122	0.0191	0.0166	0.0045	0.0081
16	3	90%	2111749.8	11723845.2	659	3214	26519	17	125.60	685.23	0.0134	0.0098	0.0060	0.0107	0.0095	0.0065	0.0050	0.0071
		80%	41433.3	148176.9	900	2762	17081	12	3.37	10.54	0.0280	0.0145	0.0080	0.0094	0.0133	0.0097	0.0037	0.0063
		70%	22073.5	74962.3	406	2102	10563	16	2.03	5.76	0.0497	0.0224	0.0073	0.0072	0.0216	0.0149	0.0074	0.0048
	5	90%	1055484.8	4641812.1	6442	14074	94299	15	116.63	462.80	0.0285	0.0170	0.0163	0.0142	0.0175	0.0113	0.0114	0.0095
		80%	123731.8	447437.6	2384	9808	30383	18	19.76	67.89	0.0352	0.0194	0.0137	0.0110	0.0177	0.0129	0.0116	0.0073
		70%	19159.7	72050.8	1136	3873	12001	11	3.19	8.67	0.0531	0.0250	0.0135	0.0121	0.0197	0.0167	0.0055	0.0081
18	3	90%	8470804.1	26263090.6	8173	46669	335005	17	451.12	1358.19	0.0140	0.0090	0.0051	0.0069	0.0077	0.0060	0.0067	0.0046
		80%	593669.8	3611399.3	2219	16609	86473	11	45.95	251.54	0.0207	0.0188	0.0079	0.0081	0.0089	0.0125	0.0046	0.0054
		70%	75422.6	211051.6	1753	11367	48150	13	7.57	18.76	0.0485	0.0235	0.0084	0.0101	0.0202	0.0157	0.0058	0.0067
	5	90%	9241667.3	24428652.2	40402	172912	2336244	19	1089.13	2811.17	0.0253	0.0170	0.0142	0.0133	0.0134	0.0113	0.0137	0.0089
		80%	449812.6	1760902.3	8801	48120	141615	13	64.60	222.66	0.0396	0.0200	0.0152	0.0120	0.0182	0.0133	0.0071	0.0080
		70%	97797.9	339614.2	3177	10124	56667	15	17.12	46.49	0.0524	0.0237	0.0120	0.0107	0.0205	0.0158	0.0090	0.0071

It is observed that the heuristic and meta-heuristic algorithms proposed in this chapter are quite accurate since all the mean error percentages are less than 0.1%. Furthermore, *GA* has the best performance and *NEH* has the worst performance. From the results of the branch-and-bound algorithm it reveals that, for the problem proposed in this chapter, it is easier to obtain the optimal solution in terms of the mean number of nodes when the learning effect strengthens. However, the standard deviation of the number of nodes exceeds its mean for all the cases, which implies that there are worst cases with a tremendous number of nodes. Therefore, the quartile of 25%, 50%, and 75% for the number of nodes is evaluated and recorded as Q1, Q2, and Q3. The observations show that the distribution for the number of nodes is right skewed because most of the mean numbers of nodes are relatively large to Q3, and it implies that most of the instances have fewer nodes. For the same instances, the box-plot of logarithm scale for the number of nodes with different parameters for the learning effect as 90%, 80%, and 70% is shown in Fig 3.1, 3.2, and 3.3, respectively. The figures illustrate that the number of nodes and the execution time grow exponentially with an increasing number of jobs.

Fig 3.1. Box-plot for logarithm scale with learning effect as 70%

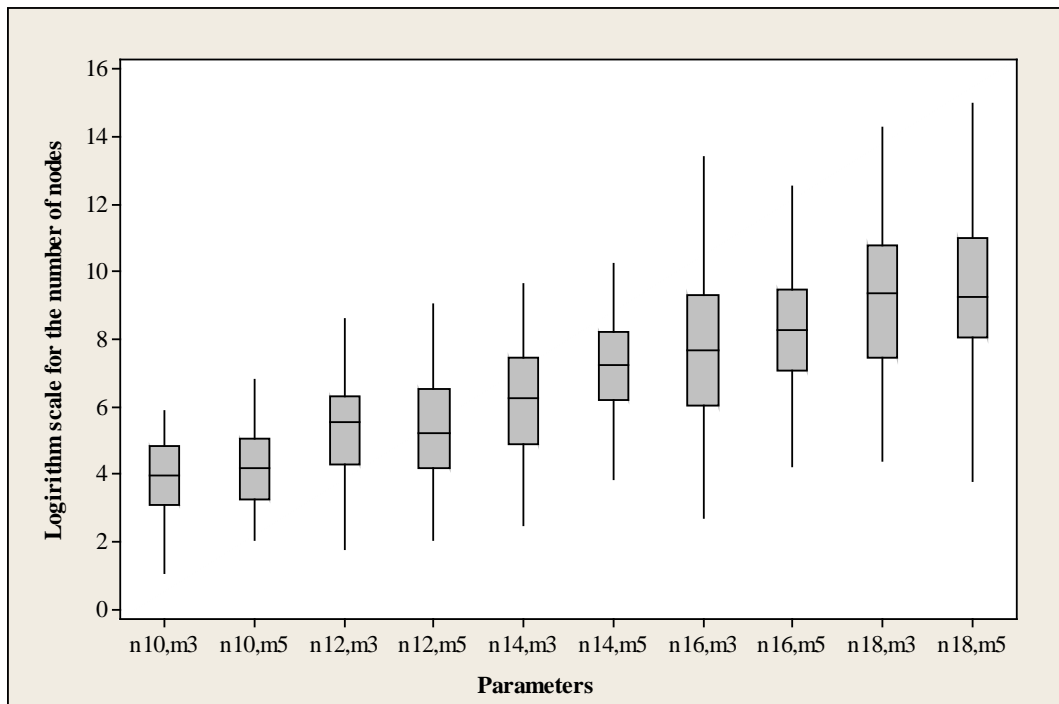


Fig 3.2. Box-plot for logarithm scale with learning effect as 80%

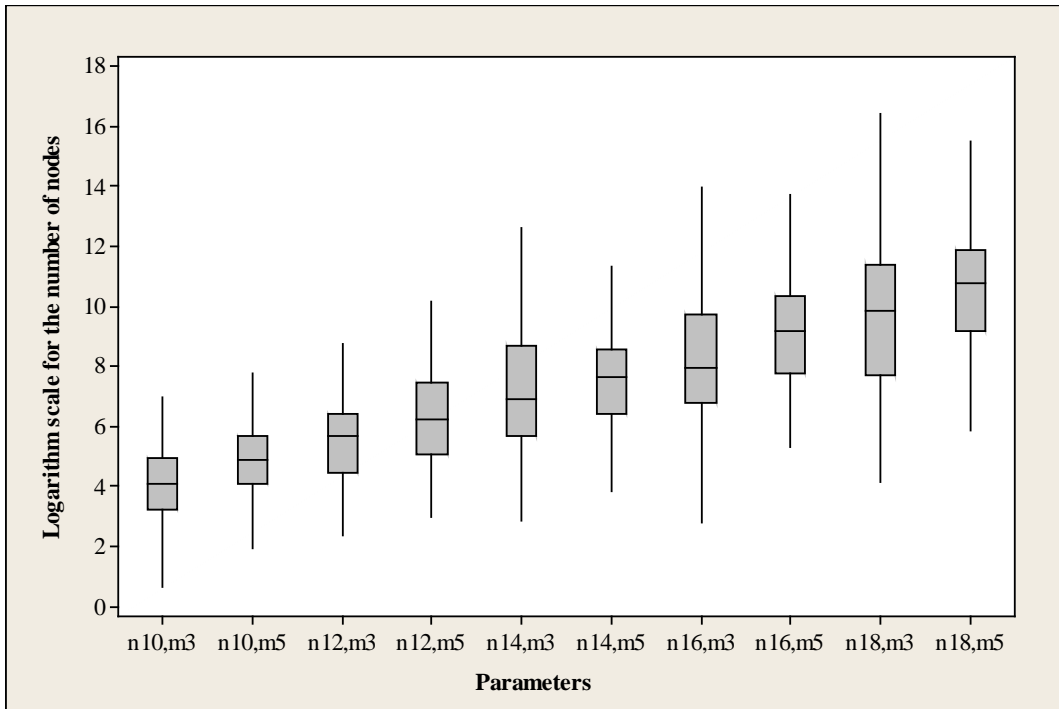
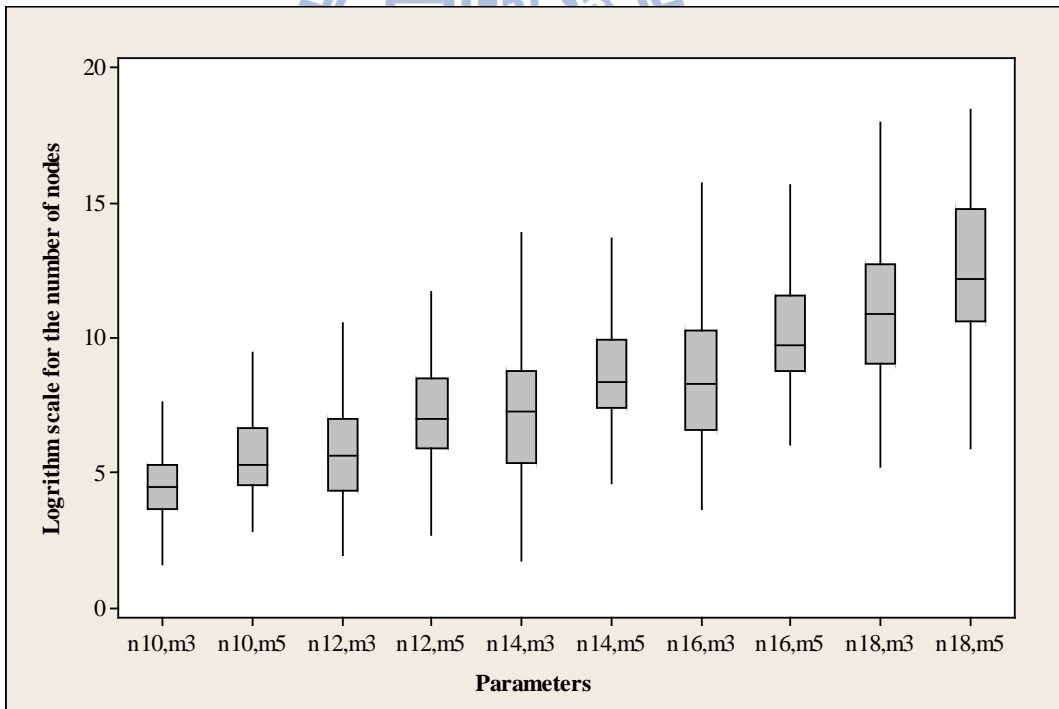


Fig 3.3. Box-plot for logarithm scale with learning effect as 90%



In order to investigate the influence of outliers, the number of outliers for each experimental condition is listed in Table 3.3, where the number of nodes for given instance which exceeds the value of $Q3 + 1.5(Q3 - Q1)$ is recorded as the outlier. The outliers are eliminated and the performance of the branch-and-bound algorithm is shown in Table 3.4.

Table 3.4. The performance of branch-and-bound algorithm of different parameters after outliers elimination

<i>n</i>	<i>m</i>	<i>a</i>	Branch and bound algorithm				
			Number of nodes		CPU times		
			Mean	s.d.	Mean	s.d.	
12	3	90%	355.9	435.2	0.022	0.026	
		80%	307.0	298.7	0.021	0.022	
		70%	268.7	252.9	0.019	0.018	
	5	90%	1912.0	2334.1	0.204	0.237	
		80%	761.0	858.6	0.090	0.089	
		70%	287.0	317.0	0.036	0.040	
	14	3	90%	2431.5	3104.2	0.195	0.234
			80%	2605.9	3474.3	0.210	0.272
			70%	801.0	964.7	0.075	0.084
5		90%	5655.6	6874.0	0.853	0.946	
		80%	2009.5	2020.1	0.328	0.307	
		70%	1800.1	1840.8	0.317	0.319	
16		3	90%	7586.8	10851.7	0.771	1.032
			80%	6814.2	9770.4	0.712	0.981
			70%	3646.5	5035.8	0.426	0.568
	5	90%	33524.6	49524.8	6.176	8.746	
		80%	10837.8	12194.7	2.415	2.769	
		70%	5519.6	6149.3	1.198	1.251	
	18	3	90%	115505.8	174775.7	11.263	16.210
			80%	36878.3	51417.1	4.025	5.316
			70%	18440.8	23462.5	2.079	2.605
5		90%	566117.6	1071722.1	82.906	144.164	
		80%	67915.4	82120.5	14.043	16.320	
		70%	20391.9	28773.9	4.521	5.776	

Table 3.4 illustrates that the means and the standard deviations for the number of nodes and execution time are all reduced by a wide margin after eliminating the outliers. Eventually, since the quantity of outliers is less than 20% of all instances for each experimental condition in this chapter, we recommend to conduct the proposed branch-and-bound algorithm for obtaining the optimal solution within a reasonable amount of time, or conduct the proposed heuristic and meta-heuristic algorithms for obtaining near-optimal solutions when the number of jobs is larger than 18.

To indicate the performance of the proposed heuristic and meta-heuristic algorithms for large job-sized problems with learning considerations, we use three different job sizes ($n= 50$, 100 and 150), four different machine sizes ($m= 5$, 10, 15, and 20) and three learning effects (90 %, 80%, and 70%) to yield the near-optimal solutions. The mean and the standard deviation of relative percentage deviation (*RPD*) are reported for each heuristic algorithm. For each instance, the *RPD* is obtained with respect to the best one of all near-optimal solutions generated by the heuristic and meta-heuristic algorithms. i.e., $RPD = \frac{V - V_{\min}}{V_{\min}}$, where V denotes the value of the makespan generated by the given heuristic or meta-heuristic algorithm and V_{\min} denotes the minimal one among the values of the makespan generated by the heuristic and meta-heuristic algorithms. Consequently, 36 experimental conditions are examined, and 100 replications are randomly generated for each condition. A total of 3,600 instances are generated and the results are listed in Table 3.5.

Table 3.5. The relative percentage deviation of heuristic algorithms

			Relative percentage deviation (RPD)								
<i>n</i>	<i>m</i>	<i>a</i>	<i>NEH</i>		<i>FL</i>		<i>SA</i>		<i>GA</i>		
			mean	s.d.	mean	s.d.	mean	s.d.	mean	s.d.	
50	5	90%	0.0479	0.0268	0.0142	0.0074	0.0133	0.0100	0.0009	0.0029	
		80%	0.0493	0.0167	0.0379	0.0130	0.0172	0.0117	0.0000	0.0004	
		70%	0.0720	0.0254	0.0654	0.0194	0.0195	0.0166	0.0005	0.0023	
	10	90%	0.0877	0.0322	0.0179	0.0113	0.0138	0.0109	0.0010	0.0027	
		80%	0.0677	0.0279	0.0413	0.0159	0.0151	0.0111	0.0003	0.0015	
		70%	0.0787	0.0235	0.0610	0.0250	0.0126	0.0112	0.0010	0.0038	
	15	90%	0.0975	0.0243	0.0185	0.0130	0.0161	0.0122	0.0012	0.0034	
		80%	0.0694	0.0247	0.0429	0.0188	0.0130	0.0096	0.0006	0.0022	
		70%	0.0766	0.0217	0.0584	0.0171	0.0117	0.0112	0.0010	0.0028	
20	90%	0.1013	0.0243	0.0204	0.0145	0.0182	0.0121	0.0006	0.0020		
	80%	0.0689	0.0212	0.0432	0.0184	0.0125	0.0112	0.0003	0.0015		
	70%	0.0727	0.0203	0.0587	0.0200	0.0079	0.0086	0.0007	0.0020		
100	5	90%	0.0350	0.0159	0.0175	0.0052	0.0106	0.0067	0.0004	0.0016	
		80%	0.0524	0.0175	0.0437	0.0107	0.0144	0.0091	0.0001	0.0012	
		70%	0.0832	0.0234	0.0747	0.0178	0.0184	0.0115	0.0001	0.0012	
	10	90%	0.0750	0.0243	0.0165	0.0079	0.0127	0.0082	0.0003	0.0011	
		80%	0.0662	0.0204	0.0466	0.0128	0.0158	0.0083	0.0000	0.0003	
		70%	0.0932	0.0193	0.0731	0.0197	0.0135	0.0098	0.0003	0.0016	
	15	90%	0.0956	0.0266	0.0182	0.0090	0.0122	0.0086	0.0001	0.0008	
		80%	0.0734	0.0210	0.0481	0.0166	0.0132	0.0082	0.0002	0.0011	
		70%	0.0916	0.0179	0.0665	0.0163	0.0117	0.0091	0.0004	0.0015	
	20	90%	0.1036	0.0200	0.0183	0.0097	0.0117	0.0083	0.0003	0.0010	
		80%	0.0760	0.0212	0.0501	0.0154	0.0121	0.0077	0.0001	0.0010	
		70%	0.0871	0.0201	0.0655	0.0193	0.0068	0.0076	0.0017	0.0046	
	150	5	90%	0.0291	0.0132	0.0197	0.0052	0.0082	0.0058	0.0005	0.0024
			80%	0.0448	0.0136	0.0477	0.0090	0.0122	0.0084	0.0006	0.0030
			70%	0.0894	0.0252	0.0774	0.0155	0.0164	0.0086	0.0001	0.0005
		10	90%	0.0655	0.0200	0.0180	0.0065	0.0101	0.0065	0.0002	0.0007
			80%	0.0653	0.0175	0.0497	0.0119	0.0121	0.0061	0.0001	0.0009
			70%	0.1006	0.0162	0.0782	0.0175	0.0141	0.0117	0.0002	0.0011
15		90%	0.0902	0.0208	0.0187	0.0064	0.0098	0.0063	0.0001	0.0010	
		80%	0.0721	0.0180	0.0495	0.0147	0.0119	0.0065	0.0001	0.0006	
		70%	0.0978	0.0156	0.0698	0.0189	0.0113	0.0082	0.0006	0.0017	
20		90%	0.0992	0.0195	0.0180	0.0068	0.0098	0.0062	0.0002	0.0009	
		80%	0.0739	0.0177	0.0515	0.0142	0.0116	0.0067	0.0000	0.0004	
		70%	0.0940	0.0175	0.0696	0.0200	0.0101	0.0082	0.0002	0.0007	

In Table 3.5, the value of *RPD* from *GA* is the minimal one among all heuristic and meta-heuristic algorithms for every experiment condition. The observation shows that *GA* is more accurate than the other three algorithms. However, as all the *RPD* values are greater than zero, it implies that there is no any algorithm which completely dominates the others. From the values of *RPD* for the heuristic and meta-heuristic algorithms, one-way analysis of variance (ANOVA) with a significance of 5% is applied to test that the mean values of *RPD* are all the same among the heuristic and meta-heuristic algorithms or whether at least one differs from the others. The results are given in Table 3.6.

Table 3.6. One-way ANOVA for *RPD* of four heuristics

Source	<i>DF</i>	<i>SS</i>	<i>MS</i>	<i>F</i>	<i>p-value</i>
Factor	3	0.124511	0.041504	199.66	0.000
Error	140	0.029101	0.000208		
Total	143	0.153612			

Since the *p*-value is below the significance level, it implies that the mean values of *RPD* are not all identical. Therefore, the efficiency among the heuristic and meta-heuristic algorithms should be considered. Furthermore, the Tukey test with a significance of 5% is implemented to compare the values of *RPD* among the heuristic and meta-heuristic algorithms. The results of Tukey test are summarized in Table 3.7.

Table 3.7. Tukey-test results of four heuristics

<i>FL</i> subtracted from:			
	Lower	Center	Upper
<i>NEH</i>	0.02331	0.03215	0.04100
<i>SA</i>	-0.04009	-0.03124	-0.02240
<i>GA</i>	-0.05249	-0.04365	-0.03481

<i>NEH</i> subtracted from:			
	Lower	Center	Upper
<i>SA</i>	-0.07224	-0.06340	-0.05455
<i>GA</i>	-0.08465	-0.07580	-0.06696

<i>SA</i> subtracted from:			
	Lower	Center	Upper
<i>GA</i>	-0.02125	-0.01241	-0.00356

The test results imply that *GA* is the best among the four algorithms, follows by *SA* and *FL*, and finally *NEH*. Thus, the proposed genetic algorithm is recommended to obtain the near-optimal solution for proposed problem in this chapter.

3.5 Summary

This chapter examines an m -machine flowshop problem with position-based learning effects where the aim is to minimize the makespan. A dominance property and a lower bound are proposed to conduct a branch-and-bound procedure for optimizing the proposed problem. In addition, this chapter also introduces the learning effect to two well-known existing heuristic and two meta-heuristic algorithms for approximating the proposed problem. The computational results show that the branch-and-bound algorithm can solve problems of up to 18 jobs within a reasonable amount of time, and demonstrate that *GA* performs best for small job-sized problems. Meanwhile, for large job-sized problems, *GA* also has identical performance. Therefore, we recommend the proposed genetic algorithm to obtain the near-optimal sequence.

Chapter 4

Bi-criteria minimization for m -machine flowshop scheduling problem with machine- and position-based learning effects

4.1 Notations and problem statement

The following notations are applied throughout this chapter.

N : Set of jobs which contains n jobs, i.e., $N = \{1, 2, \dots, n\}$.

S : Subset of N with s scheduled jobs.

U : Subset of N with $n - s$ unscheduled jobs.

m : Number of machines.

M_i : i th machine, where $i = 1, 2, \dots, m$.

J_j : Job j , where $j = 1, 2, \dots, n$.

p_{ij} : Normal processing time of J_j on M_i .

p_{ijr} : Actual processing time of J_j on M_i when J_j is scheduled at position r .

a_i : Learning index on M_i with $\forall a_i < 0$ for $i = 1, 2, \dots, m$.

[]: The symbol which denotes the job order in a sequence.

α : The weight of the objective function with $0 \leq \alpha \leq 1$.

LB : The lower bound of the objective value based on the given node.

The description of the problem with machine- and position-based learning effects in an m -machine flowshop environment is described as follows. Assume that there is a jobs set N with n jobs to be processed on m machines. Each J_j includes m operations on

corresponding machines which denoted as $O_{i,j}$ for $i=1,2,\dots,m$ and $j=1,2,\dots,n$. For the processing procedure, the starting time of $O_{i,j}$ must be the larger completion time of $O_{i-1,j}$ and $O_{i,j-1}$. In addition, the sequence of jobs is identical on all the machines. Let p_{ij} denote the normal processing time of J_j on M_i . The actual processing time p_{ijr} of J_j on M_i declines based on its position r in a sequence, i.e.,

$$p_{ijr} = p_{ij} r^{\alpha_i},$$

where $i=1,2,\dots,m$, and $j,r=1,2,\dots,n$.

The aim of this chapter is to seek a sequence for minimizing a weighted sum of the total completion time and the makespan. For a given sequence θ with n jobs, let $C_{i[r]}(\theta)$ denotes the completion time at the r th position on M_i in sequence θ . The objective of this chapter is to obtain a sequence θ^* such that

$\alpha \sum_{j=1}^n C_{m[j]}(\theta^*) + (1-\alpha)C_{m[n]}(\theta^*) \leq \alpha \sum_{j=1}^n C_{m[j]}(\theta) + (1-\alpha)C_{m[n]}(\theta)$ for any sequence θ of all permutations.

4.2 Dominance property

A rule is represented in the following theorem which distinguishing the dominance between two varied sequences concluding same jobs.

Theorem 4.1: There are two partial sequences of set N , that is $\theta_1 = (\sigma_1, \pi)$ and $\theta_2 = (\sigma_2, \pi)$,

in which σ_1 and σ_2 denotes the partial sequence of set S , and π denote a partial sequence

of set U . If $\alpha \sum_{j=1}^s [C_{m[j]}(\sigma_2) - C_{m[j]}(\sigma_1)] > [\alpha(n-s-1) + 1] \max_{1 \leq i \leq m} \{C_{i[s]}(\sigma_1) - C_{i[s]}(\sigma_2)\}$, then θ_1

dominates θ_2 .

Proof:

For $k = 1, 2, \dots, m$, we have

$$C_{k[n]}(\theta_1) = \max_{1 \leq i \leq k} \left\{ C_{i,[n-1]}(\theta_1) + \sum_{u=i}^k p_{u[n]}(n)^{a_u} \right\} = C_{i_1,[n-1]}(\theta_1) + \sum_{u=i_1}^k p_{u[n]}(n)^{a_u} \quad \text{for } 1 \leq i_1 \leq k.$$

Similarly,

$$C_{k[n]}(\theta_2) = C_{i_2,[n-1]}(\theta_2) + \sum_{u=i_2}^k p_{u[n]}(n)^{a_u} \quad \text{for } 1 \leq i_2 \leq k.$$

Then we have

$$\begin{aligned} C_{k[n]}(\theta_1) - C_{k[n]}(\theta_2) &\leq \left[C_{i_1,[n-1]}(\theta_1) + \sum_{u=i_1}^k p_{u[n]}(n)^{a_u} \right] - \left[C_{i_1,[n-1]}(\theta_2) + \sum_{u=i_1}^k p_{u[n]}(n)^{a_u} \right] \\ &\leq \max_{1 \leq i \leq m} \{ C_{i[n-1]}(\theta_1) - C_{i[n-1]}(\theta_2) \}. \end{aligned}$$

By an induction, for $k = m$, we have

$$C_{m[s+l]}(\theta_1) - C_{m[s+l]}(\theta_2) \leq \max_{1 \leq i \leq m} \{ C_{i,[s]}(\sigma_1) - C_{i,[s]}(\sigma_2) \} \dots \dots \dots (4-1)$$

, where $1 \leq l \leq n - s$.

From equation (4-1), we have

$$\begin{aligned} &\left[\alpha \sum_{j=1}^n C_{m[j]}(\theta_1) + (1 - \alpha) C_{m[n]}(\theta_1) \right] - \left[\alpha \sum_{j=1}^n C_{m[j]}(\theta_2) + (1 - \alpha) C_{m[n]}(\theta_2) \right] \\ &\leq \alpha \sum_{j=1}^s [C_{m[j]}(\sigma_1) - C_{m[j]}(\sigma_2)] + [\alpha(n - s - 1) + 1] \max_{1 \leq i \leq m} \{ C_{i,[s]}(\sigma_1) - C_{i,[s]}(\sigma_2) \} \dots \dots \dots (4-2) \end{aligned}$$

From

$$\alpha \sum_{j=1}^s [C_{m[j]}(\sigma_2) - C_{m[j]}(\sigma_1)] > [\alpha(n - s - 1) + 1] \max_{1 \leq i \leq m} \{ C_{i,[s]}(\sigma_1) - C_{i,[s]}(\sigma_2) \},$$

the value for the left side of equation (4-2) is negative and it implies θ_1 dominates θ_2 .

Therefore, we have σ_1 dominates σ_2 .

In this chapter, the theorem is simplified as the property which requires considering two adjacent jobs. The property is applied in the proposed branch-and-bound algorithm and

presented below.

Property 4.1: In set S , let σ denote a partial sequence with $s-2$ jobs, and the remained jobs are scheduled in the last two positions as J_1 and J_2 . The two sequences based on σ are represented as $S_1 = (\sigma, J_1, J_2)$ and $S_2 = (\sigma, J_2, J_1)$, and $C_{iJ_j}(S_l)$ denotes the completion time of J_j on M_i in S_l for $j, l = 1, 2$ and $i = 1, 2, \dots, m$. If

$$\alpha [C_{mJ_2}(S_2) + C_{mJ_1}(S_2) - C_{mJ_1}(S_1) - C_{mJ_2}(S_1)] > [\alpha(n-s-1) + 1] \max_{1 \leq i \leq m} \{C_{iJ_2}(S_1) - C_{iJ_1}(S_2)\},$$

then S_1 dominates S_2 .

4.3 Lower bound

In addition to dominance property, another procedure to eliminate nodes in branching tree is calculating the lower bound of the objective value. In this chapter, a lower bound is established to speed up the procedure of the proposed branch-and-bound algorithm. The lower bound is described as follows.

Let θ denote a sequence with s scheduled and $n-s$ unscheduled jobs of set N . For $1 \leq k \leq m$, the completion time of $(s+1)$ th job on M_k is as

$$\begin{aligned} C_{k[s+1]}(\theta) &= \max \{C_{k-1[s+1]}(\theta), C_{k[s]}(\theta)\} + p_{k[s+1]}(s+1)^{a_k} \\ &\geq C_{k[s]}(\theta) + p_{k[s+1]}(s+1)^{a_k}. \end{aligned}$$

Thus, the completion time of $(s+1)$ th job on M_m is presented as

$$C_{m[s+1]}(\theta) \geq C_{k[s]}(\theta) + p_{k[s+1]}(s+1)^{a_k} + \sum_{i=k+1}^m p_{i[s+1]}(s+1)^{a_i}.$$

Furthermore, the completion time of $(s+2)$ th job on M_k is as

$$C_{k[s+2]}(\theta) = \max \{C_{k-1[s+2]}(\theta), C_{k[s+1]}(\theta)\} + p_{k[s+2]}(s+2)^{a_k}$$

$$\geq C_{k[s]}(\theta) + p_{k[s+1]}(s+1)^{a_k} + p_{k[s+2]}(s+2)^{a_k}$$

Thus, the completion time of $(s+2)$ th job on M_m is as

$$C_{m[s+2]}(\theta) \geq C_{k[s]}(\theta) + \sum_{v=1}^2 p_{k[s+v]}(s+v)^{a_k} + \sum_{i=k+1}^m p_{i[s+2]}(s+2)^{a_i}.$$

By an induction, we have the underestimated value of the completion time for $(s+l)$ th job

on M_m based on M_k machine as

$$C_{k[s]}(\theta) + \sum_{v=1}^l p_{k[s+v]}(s+v)^{a_k} + \sum_{i=k+1}^m p_{i[s+l]}(s+l)^{a_i}$$

The objective function is presented as

$$\alpha \sum_{j=1}^n C_{m[j]}(\theta) + (1-\alpha)C_{m[n]}(\theta) = \alpha \sum_{j=1}^s C_{m[j]}(\theta) + \alpha \sum_{j=s+1}^{n-1} C_{m[j]}(\theta) + C_{m[n]}(\theta).$$

$$\geq \alpha \sum_{j=1}^s C_{m[j]}(\theta) + [\alpha(n-s-1)+1]C_{k[s]}(\theta) + \sum_{l=1}^{n-s} [\alpha(n-s-l)+1](s+l)^{a_k} p_{k[s+l]}$$

$$+ \sum_{l=1}^{n-s} \left\{ [\alpha + I(l)] \sum_{i=k+1}^m p_{i[s+l]}(s+l)^{a_i} \right\}$$

$$, \text{ where } I(l) = \begin{cases} 1-\alpha, & l = n-s \\ 0, & l \neq n-s \end{cases}.$$

Since $[\alpha(n-s-l)+1](s+l)^{a_k}$ decreases as l increases, then we have

$$\alpha \sum_{j=1}^n C_{m[j]}(\theta) + (1-\alpha)C_{m[n]}(\theta)$$

$$\geq \alpha \sum_{j=1}^s C_{m[j]}(\theta) + [\alpha(n-s-1)+1]C_{k[s]}(\theta) + \sum_{l=1}^{n-s} [\alpha(n-s-l)+1](s+l)^{a_k} p_{k(s+l)}$$

$$+ \sum_{l=1}^{n-s} \left\{ [\alpha + I(l)] \sum_{i=k+1}^m p_{i[s+l]}(s+l)^{a_i} \right\} \dots\dots\dots(4-3)$$

where $p_{k(s+l)}$ denotes the l th smallest normal processing time on M_k of the job in set U .

To minimize the final term of equation (4-3), a Hungarian algorithm is applied and the

matrix for it is formed as follows.

$$\begin{bmatrix} \alpha \sum_{i=k+1}^m p_{i\{s+1\}}(s+1)^{a_i} & \alpha \sum_{i=k+1}^m p_{i\{s+1\}}(s+2)^{a_i} & \cdots & \alpha \sum_{i=k+1}^m p_{i\{s+1\}}(n-1)^{a_i} & \sum_{i=k+1}^m p_{i\{s+1\}}(n)^{a_i} \\ \alpha \sum_{i=k+1}^m p_{i\{s+2\}}(s+1)^{a_i} & \alpha \sum_{i=k+1}^m p_{i\{s+2\}}(s+2)^{a_i} & \cdots & \alpha \sum_{i=k+1}^m p_{i\{s+2\}}(n-1)^{a_i} & \sum_{i=k+1}^m p_{i\{s+2\}}(n)^{a_i} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \alpha \sum_{i=k+1}^m p_{i\{n\}}(s+1)^{a_i} & \alpha \sum_{i=k+1}^m p_{i\{n\}}(s+2)^{a_i} & \cdots & \alpha \sum_{i=k+1}^m p_{i\{n\}}(n-1)^{a_i} & \sum_{i=k+1}^m p_{i\{n\}}(n)^{a_i} \end{bmatrix},$$

where $p_{i\{s+l\}}$ is the normal processing time on M_i of the jobs in set U for $1 \leq l \leq n-s$. Let

H_k denote the optimal value of the proposed Hungarian algorithm. Therefore, the

underestimated value of the objective function for θ based on M_k is as

$$\alpha \sum_{j=1}^s C_{m\{j\}}(\theta) + [\alpha(n-s-1)+1]C_{k\{s\}}(\theta) + \sum_{l=1}^{n-s} [\alpha(n-s-l)+1](s+l)^{a_k} p_{k\{s+l\}} + H_k.$$

In order to make the lower bound stricter, the underestimated value based on every machine is considered and the lower bound is obtained as

$$LB = \alpha \sum_{j=1}^s C_{m\{j\}}(\theta) + \max_{1 \leq k \leq m} \left\{ [\alpha(n-s-1)+1]C_{k\{s\}}(\theta) + \sum_{l=1}^{n-s} [\alpha(n-s-l)+1](s+l)^{a_k} p_{k\{s+l\}} + H_k \right\}$$

4.4 Computational results

In the procedure of proposed heuristic algorithms, the jobs with larger total processing time (i.e. $\sum_{i=1}^m p_{ij}$ for $j=1,2,\dots,n$) have higher priority to be selected in *NEH*, while smaller in *FL*. In addition, since the machine- and position-based learning effects are considered in this chapter, the ratios of the reduction for the actual processing time are varied on different machines. Therefore, *NEH_W* and *FL_W* are adapted from *NEH* and *FL* by utilizing the weighted total processing time (i.e. $\sum_{i=1}^m w_i p_{ij}$ for $j=1,2,\dots,n$) to determine the priority of the jobs, in which the machines with weaker learning effect have

larger weight. For example, here are three machine-based learning indices as $a_1 = -0.322$, $a_2 = -0.152$ and $a_3 = -0.515$. Then the weights are set as $w_1 = 2$, $w_2 = 3$ and $w_3 = 1$.

Several computational experiments are implemented in this chapter to assess the performance of the branch-and-bound and the heuristic algorithms. All the algorithms are coded in Fortran 90 and run on a personal computer with 2.89 GHz AMD Athlon™ II X4 635 Processor and 3.25GB RAM with Windows XP. The normal processing times of all operations are generated from a discrete uniform distribution over the integers 1 to 100. Moreover, in order to discuss the influence on the proposed algorithms for different assignments of learning effects under the same learning indices set, five learning patterns denoted as *Ran*, *Inc*, *Dec*, *SL* and *WL* are proposed and expressed as follows.

Ran: The learning effects are randomly assigned to the machines.

Inc: The stronger learning effects are assigned to the rear machines.

Dec: The weaker learning effects are assigned to the rear machines

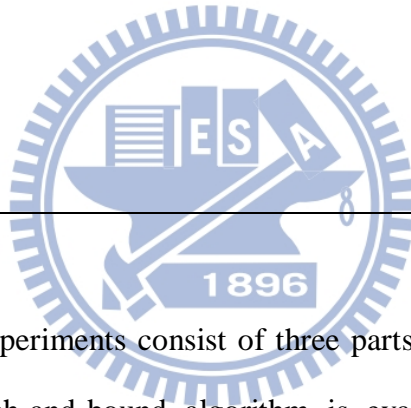
SL: The stronger learning effects are assigned to the machines with the larger value of $\sum_{j=1}^n p_{ij}$
for $i = 1, 2, \dots, m$.

WL: The weaker learning effects are assigned to the machines with the larger value of $\sum_{j=1}^n p_{ij}$
for $i = 1, 2, \dots, m$.

The learning indices set of all computational experiments in this chapter is shown in Table 4.1.

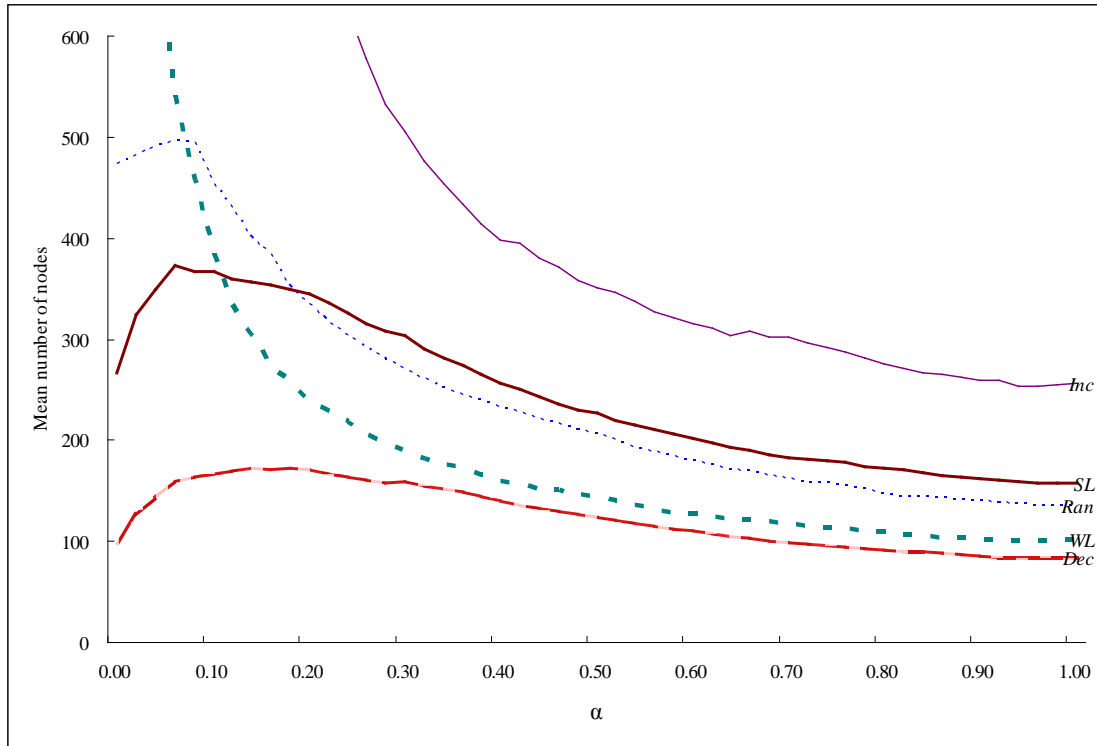
Table 4.1. The index set of the learning effects

	Number of machines			
	5	7	10	15
	-0.152	-0.152	-0.152	-0.152
	-0.234	-0.218	-0.188	-0.175
	-0.322	-0.269	-0.225	-0.199
	-0.415	-0.322	-0.263	-0.222
	-0.515	-0.377	-0.302	-0.247
		-0.434	-0.342	-0.271
		-0.515	-0.383	-0.296
learning indices			-0.426	-0.322
			-0.469	-0.348
			-0.515	-0.374
				-0.401
				-0.429
				-0.457
				-0.485
				-0.515



The computational experiments consist of three parts. In the first part, the influence of different α on the branch-and-bound algorithm is evaluated. The number of jobs and machines is set as 10 and 5, respectively. Then 100 replications are randomly generated. Consequently, a total of 100 examples are generated to be tested. In addition, 51 different α are given with values from 0 to 1 with an increment as 0.02, i.e., $\alpha = 0, 0.02, 0.04, \dots, 1$. The five learning patterns and 51 different α are considered in each example and the results are illustrated in Figs 4.1 and 4.2.

Fig 4.1. The number of nodes for the branch-and-bound algorithm under different α ($n=10$)



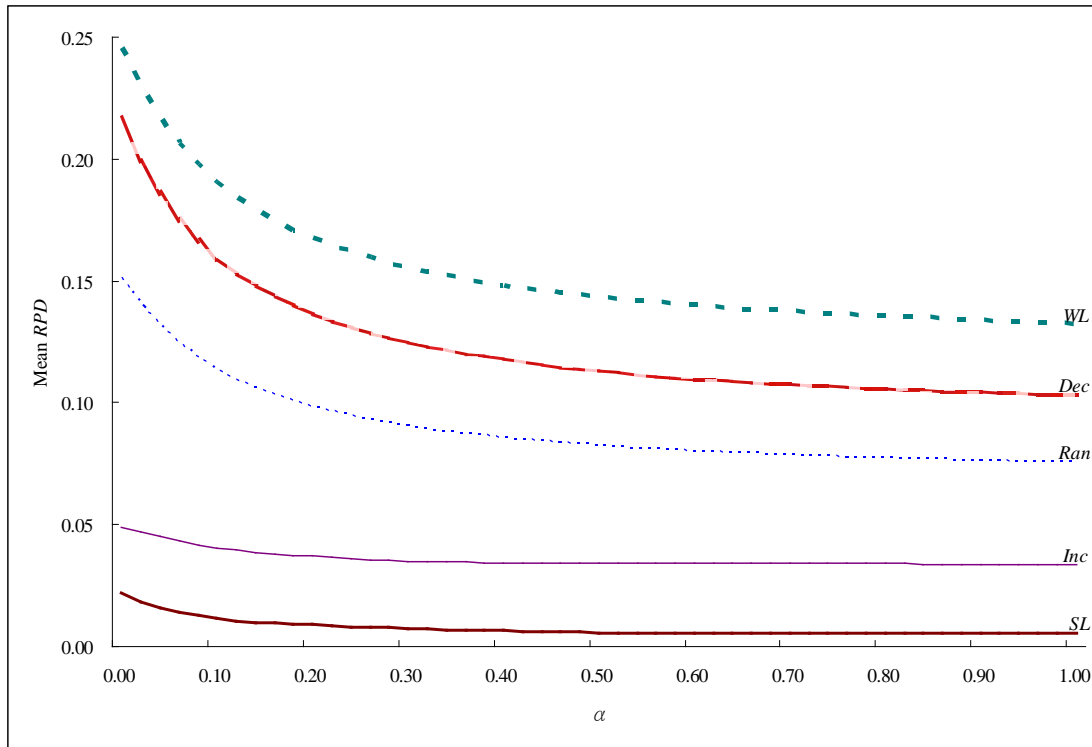
In Fig 4.1, the mean numbers of nodes for all experimental conditions are illustrated. It is observed that the problem proposed in this chapter is easier to solve as α increases with respect to the trend of the mean number of nodes. The reason is that the property and the lower bound are more efficient in the branch-and-bound algorithm with larger α . Furthermore, *Dec* is the easiest among five learning patterns for seeking the optimal solution, and *Inc* is the worst. In addition, the optimal objective values for five learning patterns are discussed. Then the relative percentage deviation for five learning patterns is denoted as RPD_o and its mean is illustrated in Fig 4.2. For each example, the RPD_o is calculated as

$$\frac{\lambda - \lambda_{\min}}{\lambda_{\min}} \times 100\%,$$

where λ denotes the optimal objective value under one of five given learning patterns, and λ_{\min} is the minimum among all λ . It is observed that the optimal objective value under *SL* is the lowest among five learning patterns, followed by *Inc*, *Ran* and *Dec*, and finally *WL*.

However, there is no determined priority among five learning patterns since all mean RPD_o are larger than zero.

Fig 4.2. The relative percentage deviation of the learning patterns for the optimal objective value under different α ($n=10$)



In the second part of the computational experiments, the numbers of jobs are set as 12, 14 and 16, and numbers of machines are set as 5 and 7. Furthermore, three α are given as 0.25, 0.50 and 0.75. Then 100 replications are randomly generated. Hence, a total of 1800 examples are generated to be tested in which the five learning patterns are considered. Then the results are listed in Tables 4.2 to 4.6.

Table 4.2. The performance of the branch-and-bound algorithm

<i>n</i>	<i>m</i>	Pattern	$\alpha = 0.25$				$\alpha = 0.50$				$\alpha = 0.75$			
			Number of nodes		Cpu times		Number of nodes		Cpu times		Number of nodes		Cpu times	
			mean	max	mean	max	mean	max	mean	max	mean	max	mean	max
12	5	<i>Ran</i>	1550.43	9213	1.84	8.34	1145.36	8605	1.55	8.63	901.69	6087	1.36	7.14
		<i>Inc</i>	3992.39	34418	2.66	19.56	2548.42	14103	1.98	10.08	1893.09	10818	1.65	7.86
		<i>Dec</i>	429.49	3109	0.40	2.09	347.98	3010	0.37	1.98	297.70	2682	0.34	1.78
		<i>SL</i>	1561.97	10346	1.01	4.20	1170.21	10062	0.88	4.02	969.24	7022	0.79	3.30
		<i>WL</i>	1045.37	11976	0.86	4.97	755.21	7559	0.73	4.58	612.87	6531	0.66	4.09
7		<i>Ran</i>	2328.46	19105	1.32	9.09	1624.33	22124	1.08	10.16	1049.45	12119	0.88	5.38
		<i>Inc</i>	6377.07	54718	5.68	25.11	3816.40	25625	4.19	20.97	2713.35	18196	3.45	17.97
		<i>Dec</i>	635.63	6673	0.89	6.38	486.78	5711	0.80	5.70	363.18	3977	0.71	5.39
		<i>SL</i>	1920.45	10152	2.03	8.16	1278.22	6932	1.67	7.67	983.96	7241	1.45	7.92
		<i>WL</i>	1277.90	10061	1.70	10.41	871.76	6145	1.38	7.09	709.46	5993	1.22	7.05
14	5	<i>Ran</i>	8397.72	97712	13.43	99.28	5727.28	56055	10.91	74.48	4845.95	46274	9.82	73.30
		<i>Inc</i>	31883.81	418776	44.54	372.50	17452.31	169405	30.27	216.80	13144.88	97435	25.12	175.41
		<i>Dec</i>	1769.26	12080	3.22	16.92	1374.09	9108	2.94	17.28	1179.42	10186	2.76	17.73
		<i>SL</i>	10382.89	115578	12.64	55.16	6689.93	49969	10.17	44.33	5140.81	28730	8.80	43.48
		<i>WL</i>	3992.79	38389	10.23	84.28	2948.51	28213	8.34	67.02	2563.70	25591	7.50	62.02

	7	<i>Ran</i>	16271.02	142636	32.37	290.52	9917.81	122468	24.74	274.36	7584.29	99124	21.15	244.52
		<i>Inc</i>	74506.94	1980840	102.24	981.52	32558.74	322750	67.46	399.08	20916.03	124587	52.36	355.20
		<i>Dec</i>	3317.76	87609	7.91	148.41	2486.66	74205	6.95	139.94	1767.69	47392	5.95	108.64
		<i>SL</i>	12229.56	86179	23.97	123.91	8137.72	42532	19.40	99.19	6565.91	35383	17.09	89.53
		<i>WL</i>	8076.68	158484	22.38	389.20	5646.80	105127	17.57	280.86	4478.99	76297	15.16	224.95
16	5	<i>Ran</i>	105044.01	2408452	349.62	4578.30	55620.78	600087	246.60	2274.17	39622.63	426218	201.16	1909.00
		<i>Inc</i>	201758.16	2061604	514.57	3209.59	120521.06	1270538	350.70	2457.27	93087.06	1411099	288.07	2618.19
		<i>Dec</i>	7520.64	107243	24.58	200.00	5155.33	41868	21.69	156.70	4148.91	37436	19.70	140.19
		<i>SL</i>	66831.54	626475	233.56	1304.53	43107.35	360252	186.24	931.81	35647.61	289344	166.05	888.14
		<i>WL</i>	31332.85	957031	103.05	1421.47	20896.89	575854	78.99	992.58	16706.67	402084	68.83	781.89
	7	<i>Ran</i>	116674.80	2455027	238.17	3462.94	69780.49	1029782	167.75	2050.94	48757.11	674012	201.16	1909.00
		<i>Inc</i>	740132.88	5188924	2252.10	16791.42	383647.16	4065895	1369.95	13054.45	263149.47	3726265	1034.44	11907.66
		<i>Dec</i>	18756.10	145112	83.65	545.95	13486.31	101862	73.19	504.02	11616.93	89494	67.83	524.33
		<i>SL</i>	105188.86	1447154	185.16	1820.66	62497.38	842215	130.71	864.78	44370.55	531122	108.13	666.53
		<i>WL</i>	55492.39	1245433	287.12	6346.81	36222.31	1007826	218.60	5101.81	29516.95	783194	188.49	4129.69

The mean and maximum number of nodes, and the mean and maximum CPU times (in seconds) of the branch-and-bound algorithm are reported in Table 4.2. It reveals that the number of nodes and the execution times increase significantly as the number of jobs or machines increases since the problem proposed in this chapter is NP-hard. The optimal solution is easier to be sought for the proposed problem with a larger α in terms of the number of nodes and CPU times. Furthermore, the problem under *Dec* is the easiest among the five learning patterns to be solved, and *Inc* is the worst. Moreover, the branch-and-bound algorithm can deal with the problems with up to 16 jobs within a reasonable amount of time.

In order to discuss the priority over five learning patterns for obtaining lower optimal objective value, the mean and maximum RPD_o are recorded for all computational conditions in Table 4.3. As shown in Table 4.3, it reveals that the optimal objective value under *SL* is the lowest among five learning patterns, follows by *Inc*, *Ran* and *Dec*, and finally *WL*. It implies that assigning the stronger learning effect to the machine with the heavier workload might obtain a lower optimal objective value.

Table 4.3. The comparison among five learning patterns for the optimal objective value

<i>n</i>	<i>m</i>	Patten	value					
			RDP_o					
			$\alpha = 0.25$		$\alpha = 0.50$		$\alpha = 0.75$	
			mean	max	mean	max	mean	max
12	5	<i>Ran</i>	0.089	0.297	0.078	0.275	0.073	0.267
		<i>Inc</i>	0.049	0.292	0.046	0.275	0.044	0.267
		<i>Dec</i>	0.125	0.356	0.109	0.324	0.103	0.310
		<i>SL</i>	0.007	0.046	0.006	0.041	0.005	0.039
		<i>WL</i>	0.161	0.346	0.144	0.316	0.137	0.303

	7	<i>Ran</i>	0.070	0.195	0.063	0.167	0.060	0.163
		<i>Inc</i>	0.025	0.119	0.024	0.105	0.024	0.103
		<i>Dec</i>	0.125	0.237	0.109	0.217	0.103	0.212
		<i>SL</i>	0.007	0.064	0.006	0.057	0.005	0.057
		<i>WL</i>	0.136	0.301	0.121	0.273	0.115	0.264

14	5	<i>Ran</i>	0.099	0.392	0.090	0.346	0.086	0.327
		<i>Inc</i>	0.054	0.200	0.050	0.197	0.048	0.194
		<i>Dec</i>	0.132	0.392	0.117	0.346	0.111	0.327
		<i>SL</i>	0.006	0.067	0.005	0.046	0.005	0.037
		<i>WL</i>	0.171	0.398	0.155	0.351	0.148	0.332

	7	<i>Ran</i>	0.083	0.327	0.075	0.296	0.071	0.288
		<i>Inc</i>	0.022	0.157	0.022	0.136	0.021	0.126
		<i>Dec</i>	0.141	0.330	0.125	0.299	0.119	0.288
		<i>SL</i>	0.006	0.056	0.005	0.039	0.004	0.038
		<i>WL</i>	0.157	0.328	0.141	0.295	0.135	0.282

16	5	<i>Ran</i>	0.095	0.271	0.086	0.247	0.083	0.239
		<i>Inc</i>	0.068	0.301	0.063	0.286	0.061	0.279
		<i>Dec</i>	0.136	0.413	0.122	0.376	0.117	0.365
		<i>SL</i>	0.006	0.060	0.005	0.054	0.005	0.052
		<i>WL</i>	0.180	0.416	0.166	0.379	0.160	0.369

	7	<i>Ran</i>	0.084	0.276	0.075	0.250	0.072	0.242
		<i>Inc</i>	0.031	0.183	0.029	0.167	0.028	0.158
		<i>Dec</i>	0.128	0.411	0.116	0.371	0.111	0.357
		<i>SL</i>	0.008	0.068	0.007	0.061	0.007	0.061
		<i>WL</i>	0.158	0.405	0.143	0.364	0.137	0.349

For the proposed heuristic algorithms, the mean and maximum error percentages under different α are reported in Tables 4.4 to 4.6. The CPU times are not presented since all heuristic algorithms for each example are executed within a second. The error percentage of the given heuristic algorithm is calculated as

$$\frac{V - V^*}{V^*} \times 100\%,$$

where V and V^* respectively denotes the near-optimal objective value yielded by the heuristic algorithm, and the optimal objective value derived by the branch-and-bound algorithm. In addition, $\min\{NEH, NEH_W\}$ denotes the better one of NEH and NEH_W for the given example, and $\min\{FL, FL_W\}$ as well denotes the better one of FL and FL_W .



Table 4.4. The performance of the heuristic algorithms ($\alpha = 0.25$)

<i>n</i>	<i>m</i>	Patten	Error percentages											
			<i>NEH</i>		<i>NEH_W</i>		$\min\{NEH,NEH_W\}$		<i>FL</i>		<i>FL_W</i>		$\min\{FL,FLW\}$	
			mean	max	mean	max	mean	max	mean	max	mean	max	mean	max
12	5	<i>Ran</i>	0.051	0.132	0.059	0.134	0.045	0.110	0.011	0.067	0.012	0.067	0.008	0.067
		<i>Inc</i>	0.033	0.097	0.037	0.105	0.029	0.097	0.010	0.056	0.012	0.054	0.007	0.045
		<i>Dec</i>	0.062	0.144	0.065	0.201	0.052	0.144	0.010	0.043	0.009	0.060	0.006	0.030
		<i>SL</i>	0.055	0.138	0.057	0.138	0.046	0.126	0.015	0.060	0.017	0.049	0.011	0.049
		<i>WL</i>	0.045	0.134	0.054	0.155	0.039	0.134	0.008	0.066	0.009	0.068	0.006	0.066
	7	<i>Ran</i>	0.046	0.141	0.052	0.150	0.040	0.097	0.015	0.054	0.014	0.058	0.010	0.054
		<i>Inc</i>	0.042	0.102	0.043	0.113	0.035	0.085	0.015	0.064	0.015	0.060	0.011	0.058
		<i>Dec</i>	0.057	0.152	0.056	0.131	0.045	0.102	0.010	0.059	0.011	0.072	0.006	0.037
		<i>SL</i>	0.051	0.140	0.051	0.115	0.041	0.110	0.014	0.049	0.016	0.066	0.011	0.039
		<i>WL</i>	0.048	0.122	0.049	0.112	0.041	0.112	0.011	0.043	0.011	0.041	0.007	0.037
14	5	<i>Ran</i>	0.050	0.110	0.058	0.136	0.045	0.095	0.012	0.044	0.011	0.065	0.008	0.040
		<i>Inc</i>	0.034	0.093	0.039	0.108	0.030	0.093	0.010	0.044	0.011	0.052	0.007	0.035
		<i>Dec</i>	0.069	0.147	0.075	0.146	0.060	0.121	0.011	0.046	0.011	0.056	0.007	0.043
		<i>SL</i>	0.058	0.152	0.063	0.152	0.049	0.120	0.016	0.057	0.019	0.065	0.011	0.037
		<i>WL</i>	0.050	0.139	0.058	0.136	0.045	0.114	0.008	0.041	0.008	0.049	0.005	0.033
	7	<i>Ran</i>	0.051	0.101	0.054	0.178	0.042	0.101	0.014	0.049	0.014	0.084	0.010	0.037
		<i>Inc</i>	0.041	0.101	0.047	0.097	0.035	0.082	0.016	0.056	0.018	0.070	0.012	0.046
		<i>Dec</i>	0.063	0.154	0.065	0.169	0.053	0.132	0.012	0.044	0.012	0.062	0.008	0.037
		<i>SL</i>	0.057	0.140	0.057	0.131	0.049	0.131	0.017	0.063	0.020	0.071	0.013	0.046
		<i>WL</i>	0.047	0.097	0.054	0.116	0.043	0.093	0.010	0.046	0.012	0.094	0.007	0.045

16	5	<i>Ran</i>	0.061	0.182	0.069	0.164	0.054	0.122	0.013	0.069	0.014	0.069	0.010	0.069
		<i>Inc</i>	0.041	0.147	0.043	0.155	0.035	0.079	0.011	0.049	0.011	0.061	0.008	0.044
		<i>Dec</i>	0.079	0.177	0.084	0.169	0.069	0.167	0.011	0.039	0.011	0.045	0.008	0.035
		<i>SL</i>	0.069	0.163	0.074	0.162	0.059	0.129	0.019	0.087	0.021	0.066	0.014	0.049
		<i>WL</i>	0.055	0.177	0.062	0.161	0.049	0.104	0.007	0.029	0.009	0.042	0.005	0.024

7	7	<i>Ran</i>	0.058	0.147	0.061	0.133	0.049	0.127	0.014	0.060	0.016	0.061	0.011	0.060
		<i>Inc</i>	0.040	0.109	0.049	0.103	0.037	0.089	0.017	0.069	0.016	0.062	0.012	0.062
		<i>Dec</i>	0.069	0.159	0.068	0.139	0.057	0.134	0.014	0.052	0.014	0.041	0.010	0.041
		<i>SL</i>	0.066	0.181	0.064	0.168	0.054	0.132	0.020	0.068	0.024	0.069	0.015	0.048
		<i>WL</i>	0.055	0.127	0.062	0.150	0.049	0.108	0.012	0.046	0.012	0.061	0.009	0.046

Table 4.5. The performance of the heuristic algorithms ($\alpha = 0.50$)

<i>n</i>	<i>m</i>	Patten	Error percentages											
			<i>NEH</i>		<i>NEH_W</i>		$\min\{NEH, NEH_W\}$		<i>FL</i>		<i>FL_W</i>		$\min\{FL, FLW\}$	
			mean	max	mean	max	mean	max	mean	max	mean	max	mean	max
12	5	<i>Ran</i>	0.056	0.153	0.063	0.146	0.049	0.128	0.010	0.065	0.012	0.064	0.008	0.064
		<i>Inc</i>	0.035	0.090	0.041	0.118	0.031	0.090	0.011	0.047	0.011	0.049	0.008	0.043
		<i>Dec</i>	0.066	0.173	0.069	0.167	0.056	0.167	0.009	0.051	0.009	0.049	0.006	0.041
		<i>SL</i>	0.049	0.142	0.054	0.138	0.042	0.138	0.013	0.049	0.014	0.048	0.010	0.048
		<i>WL</i>	0.049	0.145	0.055	0.154	0.043	0.128	0.006	0.033	0.008	0.068	0.004	0.025

7	7	<i>Ran</i>	0.050	0.137	0.051	0.153	0.041	0.106	0.015	0.060	0.013	0.063	0.010	0.046
		<i>Inc</i>	0.042	0.106	0.042	0.111	0.035	0.085	0.014	0.053	0.013	0.051	0.010	0.049
		<i>Dec</i>	0.063	0.181	0.057	0.136	0.048	0.118	0.010	0.050	0.010	0.050	0.007	0.050
		<i>SL</i>	0.052	0.137	0.051	0.130	0.042	0.125	0.013	0.051	0.015	0.049	0.010	0.037

		<i>WL</i>	0.050	0.137	0.052	0.134	0.042	0.120	0.012	0.069	0.012	0.074	0.009	0.060
14	5	<i>Ran</i>	0.056	0.124	0.062	0.161	0.048	0.104	0.013	0.056	0.012	0.056	0.008	0.047
		<i>Inc</i>	0.038	0.093	0.043	0.137	0.033	0.086	0.012	0.045	0.011	0.045	0.009	0.045
		<i>Dec</i>	0.079	0.169	0.085	0.186	0.068	0.156	0.011	0.044	0.011	0.064	0.007	0.028
		<i>SL</i>	0.058	0.129	0.066	0.151	0.051	0.122	0.016	0.064	0.019	0.082	0.012	0.055
		<i>WL</i>	0.056	0.139	0.063	0.161	0.048	0.105	0.008	0.044	0.008	0.065	0.005	0.037

	7	<i>Ran</i>	0.051	0.117	0.055	0.115	0.044	0.115	0.012	0.043	0.014	0.054	0.010	0.037
		<i>Inc</i>	0.041	0.116	0.046	0.107	0.036	0.090	0.016	0.063	0.016	0.063	0.012	0.063
		<i>Dec</i>	0.068	0.173	0.067	0.173	0.056	0.161	0.010	0.054	0.012	0.051	0.008	0.043
		<i>SL</i>	0.058	0.138	0.056	0.144	0.047	0.138	0.016	0.049	0.018	0.069	0.013	0.049
		<i>WL</i>	0.051	0.111	0.055	0.116	0.045	0.095	0.011	0.073	0.011	0.067	0.008	0.067

16	5	<i>Ran</i>	0.067	0.203	0.072	0.185	0.058	0.135	0.013	0.063	0.014	0.073	0.010	0.060
		<i>Inc</i>	0.047	0.139	0.047	0.161	0.039	0.093	0.012	0.072	0.013	0.051	0.009	0.051
		<i>Dec</i>	0.088	0.196	0.090	0.167	0.076	0.167	0.013	0.059	0.011	0.068	0.008	0.047
		<i>SL</i>	0.071	0.164	0.075	0.172	0.060	0.131	0.019	0.074	0.021	0.147	0.014	0.047
		<i>WL</i>	0.061	0.196	0.068	0.161	0.054	0.117	0.008	0.038	0.008	0.041	0.005	0.025

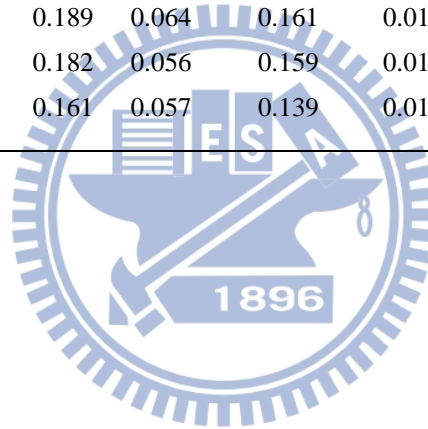
	7	<i>Ran</i>	0.061	0.150	0.065	0.148	0.053	0.144	0.014	0.061	0.014	0.048	0.010	0.046
		<i>Inc</i>	0.044	0.095	0.048	0.098	0.038	0.088	0.016	0.057	0.014	0.062	0.011	0.057
		<i>Dec</i>	0.074	0.157	0.072	0.179	0.062	0.157	0.013	0.048	0.013	0.053	0.008	0.038
		<i>SL</i>	0.066	0.146	0.065	0.177	0.055	0.146	0.019	0.059	0.019	0.073	0.013	0.040
		<i>WL</i>	0.061	0.143	0.069	0.151	0.056	0.133	0.012	0.058	0.013	0.052	0.008	0.039

Table 4.6. The performance of the heuristic algorithms ($\alpha = 0.75$)

<i>n</i>	<i>m</i>	Patten	Error percentages											
			<i>NEH</i>		<i>NEH_W</i>		$\min\{NEH,NEH_W\}$		<i>FL</i>		<i>FL_W</i>		$\min\{FL,FLW\}$	
			mean	max	mean	max	mean	max	mean	max	mean	max	mean	max
12	5	<i>Ran</i>	0.058	0.159	0.064	0.157	0.050	0.132	0.012	0.064	0.011	0.044	0.008	0.044
		<i>Inc</i>	0.036	0.086	0.039	0.092	0.031	0.081	0.011	0.044	0.011	0.053	0.008	0.035
		<i>Dec</i>	0.069	0.149	0.070	0.170	0.057	0.122	0.009	0.052	0.009	0.044	0.006	0.030
		<i>SL</i>	0.056	0.179	0.057	0.144	0.047	0.144	0.013	0.046	0.016	0.055	0.010	0.041
		<i>WL</i>	0.052	0.146	0.058	0.187	0.045	0.114	0.006	0.050	0.007	0.059	0.005	0.038
	7	<i>Ran</i>	0.050	0.137	0.052	0.133	0.042	0.109	0.013	0.051	0.013	0.046	0.009	0.043
		<i>Inc</i>	0.042	0.104	0.041	0.091	0.034	0.081	0.013	0.064	0.012	0.051	0.010	0.051
		<i>Dec</i>	0.067	0.191	0.057	0.137	0.050	0.129	0.009	0.040	0.009	0.041	0.006	0.037
		<i>SL</i>	0.054	0.143	0.052	0.132	0.042	0.130	0.014	0.060	0.015	0.045	0.010	0.043
		<i>WL</i>	0.053	0.132	0.051	0.145	0.043	0.130	0.011	0.057	0.011	0.047	0.008	0.044
14	5	<i>Ran</i>	0.057	0.131	0.064	0.131	0.050	0.109	0.013	0.054	0.012	0.050	0.008	0.045
		<i>Inc</i>	0.039	0.103	0.043	0.105	0.034	0.083	0.012	0.040	0.013	0.059	0.009	0.040
		<i>Dec</i>	0.081	0.180	0.082	0.199	0.068	0.159	0.011	0.048	0.011	0.053	0.007	0.044
		<i>SL</i>	0.060	0.137	0.066	0.154	0.050	0.127	0.017	0.084	0.018	0.057	0.012	0.043
		<i>WL</i>	0.058	0.146	0.063	0.140	0.049	0.112	0.009	0.061	0.009	0.061	0.006	0.061
	7	<i>Ran</i>	0.051	0.126	0.056	0.152	0.044	0.126	0.013	0.047	0.013	0.050	0.010	0.042
		<i>Inc</i>	0.043	0.120	0.045	0.103	0.036	0.103	0.017	0.060	0.016	0.058	0.013	0.056
		<i>Dec</i>	0.068	0.180	0.068	0.178	0.056	0.167	0.010	0.045	0.011	0.067	0.007	0.045
		<i>SL</i>	0.058	0.126	0.057	0.151	0.047	0.125	0.016	0.053	0.019	0.065	0.012	0.053
		<i>WL</i>	0.053	0.118	0.055	0.119	0.045	0.100	0.011	0.073	0.011	0.047	0.007	0.047

16	5	<i>Ran</i>	0.069	0.222	0.071	0.173	0.059	0.153	0.016	0.091	0.014	0.056	0.010	0.042
		<i>Inc</i>	0.048	0.151	0.048	0.164	0.040	0.094	0.012	0.064	0.013	0.039	0.008	0.035
		<i>Dec</i>	0.089	0.207	0.090	0.179	0.075	0.149	0.011	0.043	0.011	0.068	0.008	0.032
		<i>SL</i>	0.072	0.169	0.074	0.172	0.060	0.144	0.020	0.076	0.022	0.160	0.014	0.076
		<i>WL</i>	0.064	0.207	0.071	0.167	0.056	0.126	0.010	0.076	0.010	0.042	0.006	0.037

7		<i>Ran</i>	0.063	0.157	0.065	0.155	0.054	0.151	0.014	0.046	0.015	0.055	0.010	0.044
		<i>Inc</i>	0.045	0.097	0.046	0.110	0.038	0.094	0.015	0.054	0.016	0.067	0.011	0.043
		<i>Dec</i>	0.076	0.161	0.075	0.189	0.064	0.161	0.013	0.042	0.013	0.053	0.009	0.035
		<i>SL</i>	0.067	0.159	0.065	0.182	0.056	0.159	0.018	0.074	0.020	0.074	0.014	0.074
		<i>WL</i>	0.063	0.150	0.070	0.161	0.057	0.139	0.014	0.055	0.014	0.054	0.010	0.035



As shown in Tables 4.4 to 4.6, it is observed that all heuristic algorithms proposed in this chapter are quite accurate since the error percentages are all less than 0.1%. For evaluating the influence on the performance of the heuristic algorithms, several two-way analysis of variance (ANOVA) with a significance of 5% of the mean error percentage under each heuristic algorithm are conducted and the results are reported in Table 4.7.

Table 4.7. Two-way ANOVA of the error percentages for all heuristic algorithms

Heuristic algorithm	Source	DF	SS	MS	F	p-value
<i>NEH</i>	α	2	0.0004311	0.0002155	5.08	0.009
	Learning patterns	4	0.0089166	0.0022292	52.50	0.000
	Interaction	8	0.0001122	0.0000140	0.33	0.952
	Error	75	0.0031847	0.0000425		
	Total	89	0.0126446			
<i>NEH_W</i>	α	2	0.0001460	0.0000730	1.21	0.304
	Learning patterns	4	0.0735550	0.0018389	30.46	0.000
	Interaction	8	0.0000621	0.0000078	0.13	0.998
	Error	75	0.0045283	0.0000604		
	Total	89	0.0120920			
min{ <i>NEH,NEH_W</i> }	α	2	0.0001948	0.0000974	2.43	0.095
	Learning patterns	4	0.0056230	0.0014058	35.04	0.000
	Interaction	8	0.0000657	0.0000082	0.20	0.989
	Error	75	0.0030085	0.0000401		
	Total	89	0.0088921			
<i>FL</i>	α	2	0.0000008	0.0000004	0.08	0.919
	Learning patterns	4	0.0004772	0.0001193	25.31	0.000
	Interaction	8	0.0000074	0.0000009	0.20	0.991
	Error	75	0.0003535	0.0000047		
	Total	89	0.0008389			
<i>FL_W</i>	α	2	0.0000075	0.0000039	0.90	0.412
	Learning patterns	4	0.0007590	0.0001898	43.70	0.000
	Interaction	8	0.0000071	0.0000009	0.20	0.989
	Error	75	0.0003257	0.0000043		
	Total	89	0.0010996			
min{ <i>FL,FL_W</i> }	α	2	0.0000002	0.0000001	0.03	0.970
	Learning patterns	4	0.0003397	0.0000849	33.00	0.000
	Interaction	8	0.0000030	0.0000004	0.14	0.997
	Error	75	0.0001930	0.0000026		
	Total	89	0.0005358			

As shown in Table 4.7, it is observed that α doesn't have a significant effect on the accuracy for all heuristic algorithms except *NEH*. Then it is shown in Tables 4.4 to 4.6 that the mean error percentage of *NEH* descends as α decreases, and the reason is that the *NEH* is initially devoted to solving the makespan problem. Furthermore, it reveals that the learning pattern has a significant effect on the accuracy for all proposed heuristic algorithms. A close observation of Tables 4.4 to 4.6 shows that *Inc* is the most accurate under *NEH*, *NEH_W* and $\min\{NEH, NEH_W\}$, and *Dec* is the least accurate. Meanwhile, *SL* is the most accurate under *FL*, *FL_W* and $\min\{FL, FL_W\}$, and *WL* is the least. In addition, there is no interaction between α and the learning patterns for all heuristic algorithms. Moreover, it is shown that $\min\{NEH, NEH_W\}$ is more accurate than *NEH* and *NEH_W*, and $\min\{FL, FL_W\}$ is more accurate than *FL* and *FL_W*. It implies that there is no priority between two methods of index development utilized in the proposed heuristic algorithms. Eventually, $\min\{FL, FL_W\}$ is the most accurate among all heuristic algorithms, followed by *FL* and *FL_W*, $\min\{NEH, NEH_W\}$, and finally *NEH* and *NEH_W*.

In the last part of the computational experiments, the examples with large size of jobs are generated to perform the heuristic algorithms proposed in this chapter. Let α be set as 0.50 since most of the proposed heuristic algorithms are not affected by α for the statistical analysis in Table 4.7. Additionally, the numbers of jobs are set as 50 and 100, and numbers of machines are set as 10 and 15. Then 100 replications are randomly generated. A total of 400 examples are generated to be tested in which five learning patterns are considered in each example. Consequently, the relative percentage deviation for all heuristic algorithms is denoted as RPD_H , and its mean and maximum values are listed in Table 4.8. For each example, the RPD_H is calculated as

$$\frac{\mu - \mu_{\min}}{\mu_{\min}} \times 100\%$$

where μ denotes the near-optimal objective value for given one of all heuristic algorithms, and μ_{\min} is the minimum among all μ . As shown in Table 4.8 that FL and FL_W are both better than $\min\{NEH, NEH_W\}$ in terms of the RPD_H . It implies that the heuristic algorithm proposed by Framinan and Leisten [11] is more proper than the algorithm proposed by Nawaz et al. [23] to obtain the near-optimal solution for the problem proposed in this chapter. Finally, it is observed that $\min\{FL, FL_W\}$ is the most accurate of all proposed heuristic algorithms because of that the RDP_H are all zero. Therefore, $\min\{FL, FL_W\}$ is recommended to yield the near-optimal schedule for the problem proposed in this chapter.



Table 4.8. The comparison of the heuristic algorithms for large job-sized problem ($\alpha = 0.50$)

<i>n</i>	<i>m</i>	Pattern	RPD_H											
			<i>NEH</i>		<i>NEH W</i>		$\min\{NEH, NEH W\}$		<i>FL</i>		<i>FL W</i>		$\min\{FL, FLW\}$	
			mean	max	mean	max	mean	max	mean	max	mean	max	mean	max
50	10	<i>Ran</i>	0.072	0.143	0.073	0.133	0.064	0.106	0.004	0.053	0.004	0.030	0.000	0.008
		<i>Inc</i>	0.043	0.078	0.047	0.083	0.039	0.074	0.003	0.035	0.003	0.035	0.000	0.000
		<i>Dec</i>	0.095	0.137	0.075	0.129	0.073	0.128	0.004	0.037	0.003	0.026	0.000	0.000
		<i>SL</i>	0.073	0.135	0.071	0.117	0.064	0.117	0.004	0.027	0.007	0.047	0.000	0.000
		<i>WL</i>	0.070	0.146	0.067	0.119	0.061	0.119	0.004	0.027	0.003	0.024	0.000	0.000
15	15	<i>Ran</i>	0.062	0.127	0.060	0.114	0.053	0.108	0.006	0.038	0.003	0.028	0.000	0.000
		<i>Inc</i>	0.037	0.077	0.040	0.072	0.033	0.068	0.005	0.029	0.004	0.035	0.000	0.004
		<i>Dec</i>	0.078	0.135	0.061	0.123	0.059	0.123	0.004	0.021	0.003	0.022	0.000	0.000
		<i>SL</i>	0.065	0.107	0.059	0.107	0.054	0.106	0.005	0.029	0.004	0.024	0.000	0.000
		<i>WL</i>	0.063	0.143	0.060	0.108	0.054	0.108	0.004	0.030	0.002	0.026	0.000	0.000
100	10	<i>Ran</i>	0.080	0.133	0.081	0.136	0.072	0.116	0.004	0.028	0.004	0.034	0.000	0.000
		<i>Inc</i>	0.055	0.090	0.057	0.091	0.052	0.079	0.003	0.023	0.002	0.024	0.000	0.000
		<i>Dec</i>	0.102	0.155	0.079	0.129	0.078	0.129	0.004	0.019	0.002	0.013	0.000	0.000
		<i>SL</i>	0.089	0.133	0.087	0.132	0.082	0.132	0.003	0.018	0.004	0.029	0.000	0.000
		<i>WL</i>	0.075	0.137	0.077	0.130	0.069	0.117	0.004	0.020	0.002	0.024	0.000	0.000
15	15	<i>Ran</i>	0.073	0.132	0.068	0.109	0.065	0.103	0.005	0.047	0.002	0.021	0.000	0.000
		<i>Inc</i>	0.048	0.079	0.052	0.083	0.045	0.071	0.004	0.029	0.002	0.018	0.000	0.000
		<i>Dec</i>	0.093	0.135	0.067	0.113	0.067	0.113	0.004	0.021	0.002	0.015	0.000	0.000
		<i>SL</i>	0.073	0.117	0.068	0.105	0.063	0.102	0.004	0.022	0.003	0.022	0.000	0.000
		<i>WL</i>	0.071	0.120	0.068	0.121	0.064	0.105	0.005	0.029	0.002	0.025	0.000	0.000

4.5 Summary

In this chapter, an m -machine flowshop scheduling problem with machine- and position-based learning effects is studied to minimize the weighted sum of the total completion time and the makespan. The branch-and-bound algorithm incorporated with a dominance property and a lower bound is proposed to seek the optimal sequence, and four heuristic algorithms are established to yield the near-optimal sequences. As shown in the computational results, the proposed problem can be dealt with up to 16 jobs within a reasonable amount of time for seeking the optimal sequence. When the learning pattern is set as *Inc*, or if α is smaller, the proposed problem is harder to search for the optimal sequence by implementing the proposed branch-and-bound algorithm. Furthermore, the performances of all proposed heuristic algorithms are accurate and $\min\{FL, FL_W\}$ is recommended to obtain the near-optimal sequence. Finally, the issue for allocating the learning effects to the machines is discussed in this chapter, and it is shown that assigning the stronger learning effects to the machines with the heavier workload might obtain the better result, and it can be utilized as an important course for decision making in the scheduling field, such as assigning the operators to the machines.

Chapter 5

Concluding remarks

5.1. Conclusion

In this dissertation, two m -machine flowshop problems with position-based learning effect are studied. For each problem, a dominance property and a lower bound are proposed to conduct a branch-and-bound algorithm for obtaining the optimal sequences. In addition, because searching the optimal sequence for large job-sized problem is time consuming, this dissertation introduces learning effect into two well-known existing heuristic and two meta-heuristic algorithms to obtain the near-optimal sequences. Then the optimal sequence for small job-sized problems is utilized to assess the accuracy of the proposed heuristic and meta-heuristic algorithms. The computational experiment shows that

- Assigning the stronger learning effects to the machines with the heavier workload might obtain the better result.
- The optimal solution for the traditional flowshop scheduling problem is no longer optimal when the learning effect exists in the production environment.
- The branch-and-bound algorithm can solve problems of up to 18 jobs within a reasonable amount of time.
- We recommend to conduct the proposed branch-and-bound algorithm for obtaining the optimal sequence within a reasonable amount of time, or conduct the proposed heuristic and meta-heuristic algorithms for obtaining near-optimal sequences when the number of jobs is larger than 18.
- The heuristic and meta-heuristic algorithms proposed in this dissertation are quite accurate since all the mean error percentages are less than 0.1%.
- *GA* is recommended to derive the near-optimal sequence when the execution time is not

considered. Otherwise, the $\min\{FL, FL_W\}$ is recommended.

- The efficiency of the branch-and-bound algorithm enhances while the machines have stronger learning effects or a decreasing trend of learning effects.
- The number of nodes and the execution time grow exponentially with an increasing number of jobs because of the proposed problems are NP-hard problems.

5.2 Suggestions for further studies

Some possible suggestion could be investigated for further studies and listed as follows.

- The actual processing time of the job in the proposed model could be divided into two parts, those are the setup time with the learning consideration and the normal processing time without the learning consideration, in which the setup time is operated by the worker, and the normal processing time is operated by the machine.
- Other objective functions could be discussed, like minimizing total tardiness, minimizing the number of tardy jobs, and so on.
- The concept of multiple-agent could be introduced into the proposed scheduling problems.
- Developing the constructive heuristic algorithms or meta-heuristic algorithms to derive better near-optimal sequence.
- Searching for more practical learning model.
- The sum-of-processing- time-based learning effect could be studied.

References

- [1] Biskup, D., "Single-machine scheduling with learning considerations", European Journal of Operational Research, 115, 173-178, 1999.
- [2] Biskup, D., "A state-of-the-art review on scheduling with learning effect", European Journal of Operational Research, 188, 315-329, 2008.
- [3] Chen, P., Wu, C.C., and Lee, W.C., "A bi-criteria two-machine flowshop scheduling problem with a learning effect", The Journal of Operational Research Society, 57, 1113-1125, 2006.
- [4] Cheng, T.C.E., Cheng, S.R., Wu, W.H., Hsu, P.H., and Wu, C.C., "A two-agent single-machine scheduling problem with truncated sum-of-processing-times-based learning consideration", Computers & Industrial Engineering, 60, 534-541, 2011.
- [5] Cheng, T.C.E., Lai, P.J., Wu, C.C., and Lee, W.C., "Single-machine scheduling with sum-of-logarithm-processing-times-based learning considerations", Information Sciences, 179, 3127-3135, 2009.
- [6] Cheng, T.C.E., Wu, C.C., and Lee, W.C., "Some scheduling problems with sum-of-processing-times-based and job-position-based learning effects", Information Sciences, 178, 2476-2487, 2008.
- [7] Chung, C.S., Flynn, J., and Kirca, Ö., "A branch-and-bound algorithm to minimize the total flow time for m -machine permutation flowshop problems", International Journal of Production Economics, 79, 185-196, 2002.
- [8] Chung, C.S., Flynn, J., and Kirca, Ö., "A branch and bound algorithm to minimize the total tardiness for m -machine permutation flowshop problems", European Journal of Operational Research, 174, 1-10, 2006.
- [9] Eren, T., and Guner, E., "A bicriteria parallel machine scheduling with a learning

- effect”, International Journal of Advanced Manufacturing Technology, 40, 1202-1205, 2009.
- [10] Framinan, J.M., Gupta J.N.D., and Leisten, R., “A review and classification of heuristics for permutation flow-shop scheduling with makespan objective”, Journal of the Operational Research Society, 55, 1243-1255, 2004.
- [11] Framinan, J.M., and Leisten, R., “An efficient constructive heuristic for flowtime minimization in permutation flow shops”, OMEGA, 31, 311-317, 2003.
- [12] Garey, M.R., Johnson, D.S., and Sethi, R., “The complexity of flowshop and jobshop scheduling”, Mathematics of Operations Research, 1, 117-129, 1976.
- [13] Huang, X., Wang, M.Z., and Wang, J.B., “Single-machine scheduling with both learning effects and deteriorating jobs”, Computers & Industrial Engineering, 60, 750-754, 2011.
- [14] Janiak, A., and Rudek, R., “A new approach to the learning effect: Beyond the learning curve restrictions”, Computers and Operations Research, 35, 3727- 3736, 2008.
- [15] Janiak, A., and Rudek, R., “Experience based approach to scheduling problems with the learning effect”, IEEE Transactions on System, Man, and Cybernetics, Part A: Systems and Humans, 39, 344-357, 2009.
- [16] Janiak, A., and Rudek, R., “A note on a makespan minimization problem with a multi-ability learning effect”, Omega, 38, 213-217, 2010.
- [17] Koulamas, C. and Kyparisis, G.J., “Single-machine and two-machine flowshop scheduling with general learning function”, European Journal of Operational Research, 178, 402-407, 2007.
- [18] Koulamas, C., “A note on single-machine scheduling with job-dependent learning effects”, European Journal of Operational Research, 207, 1142-1143, 2010.
- [19] Lai, P.J., and Lee, W.C., “Single-machine scheduling with general sum-of-processing-time-based and position-based learning effects”, Omega, 39,

467-471, 2011.

- [20] Lee, W.C., and Lai, P.J., “Scheduling problems with general effects of deterioration and learning”, Information Sciences, 181, 1164-1170, 2011.
- [21] Lee, W.C., and Wu, C.C., “Minimizing total completion time in a two-machine flowshop with a learning effect”, International Journal of Production Economics, 88, 85-93, 2004.
- [22] Lee, W.C., and Wu, C.C., “Some single-machine and m -machine flowshop scheduling problems with learning considerations”. Information Sciences, 179, 3885-3892, 2009.
- [23] Lee, W.C., Wu, C.C., and Hsu, P.H., “A single-machine learning effects scheduling problem with release times”, Omega, 38, 3-11, 2010.
- [24] Lenstra, J.K., Rinnooy Kan, A.H.G., and Brucker, P., “Complexity of machine scheduling problems”, Annals of Discrete Mathematics, 1, 343-362, 1977.
- [25] Li, D.C., Hsu, P.H., Wu, C.C., and Cheng, T.C.E., “Two-machine flowshop scheduling with truncated learning to minimize the total completion time”, Computers & Industrial Engineering, In Press, 2011, doi: 10.1016/j.cie.2011.04.021
- [26] Liu, S., and Ong, H.L., “A comparative study of algorithms for the flowshop scheduling problem”, Asia-Pacific Journal of Operational Research, 19, 205-222, 2002.
- [27] Nawaz, M., Ensore, E.E., and Ham, I., “A heuristic algorithm for the m -machine, n -job flow-shop sequencing problem”, OMEGA, 11, 91-95, 1983.
- [28] Mosheiov, G., and Sidney J.B., “Scheduling with general job-dependent learning curves”, European Journal of Operational Research, 147, 665-670, 2003.
- [29] Pinedo, M., Scheduling: theory, algorithms, and systems, Second Edition, Prentice-Hall, Upper Saddle River, New Jersey, 2002.
- [30] Rajendran, C., and Ziegler, H., “An efficient heuristic for scheduling in a flowshop to minimize total weighted flowtime of jobs”, European Journal of Operational Research, 103, 129-138, 1997.

- [31] Ruiz, R., and Maroto, C., “A comprehensive review and evaluation of permutation flowshop heuristics”, European Journal of Operational Research, 165, 479-494, 2005.
- [32] Smith, W.E., “Various optimizers for single state production”, Naval Research Logistics Quarterly, 3, 59-66, 1956.
- [33] Toksari, M.D., “A branch and bound algorithm for minimizing makespan on a single machine with unequal release times under learning effect and deteriorating jobs”, Computers and Operations Research, 38, 1361- 1365, 2011.
- [34] Toksari, M.D., and Guner, E., “Parallel machine earliness/tardiness scheduling problem under the effects of position based learning and linear/nonlinear deterioration”, Computers and Operations Research, 36, 2394-2417, 2009.
- [35] Wang, J.B., and Li, J.X., “Single machine past-sequence-dependent setup times scheduling with general position-dependent and time-dependent learning effects”, Applied Mathematical Modelling, 35, 1388-1395, 2011.
- [36] Wang, L., Pan, Q.K., and Tasgetiren, M.F., “A hybrid harmony search algorithm for the blocking permutation flow shop scheduling problem”, Computers & Industrial Engineering, 61, 76-83, 2011.
- [37] Wang, J.B., Sun, L., and Sun, L., “Single machine scheduling with a learning effect and discounted costs”, International Journal of Advanced Manufacturing Technology, 49, 1141-1149, 2010.
- [38] Wang, J.B., and Wang, J.J., “Single-machine scheduling jobs with exponential learning functions”, Computers & Industrial Engineering, 60, 755-759, 2011.
- [39] Wang, X.R., Wang, J.B., Gao, W.J., and Huang, X., “Scheduling with past-sequence-dependent setup times and learning effects on single machine”, International Journal of Advanced Manufacturing Technology, 48, 739-746, 2010.
- [40] Wang, L.Y., Wang, J.B., Gao, W.J., Huang, X., and Feng, E.M., “Two single-machine scheduling problems with the effects of deterioration and learning”, International

Journal of Advanced Manufacturing Technology, 46, 715-720, 2010.

- [41] Wang, L.Y., Wang, J.B., Wang, D., Yin, N., Huang, X., and Feng, E.M., “Single-machine scheduling with a sum-of-processing-time based learning effect and deteriorating jobs”, International Journal of Advanced Manufacturing Technology, 45, 336-340, 2009.
- [42] Wang, J.B., and Xia, Z.Q., “Flow-shop scheduling with a learning effect”, The Journal of Operational Research Society, 56, 1325-1330, 2005.
- [43] Woo, H.S., and Yim, D.S., “A heuristic algorithm for mean flowtime objective in flowshop scheduling”, Computers and Operations Research, 25, 175-182, 1998.
- [44] Wright, T.P., “Factors affecting the cost of airplanes”, Journal of Aeronautical Sciences, 3, 122-128, 1936.
- [45] Wu, C.C., Hsu, P.H., Chen, J.C., Wang, N.S., and Wu, W.H., “Branch-and-bound and simulated annealing algorithms for a total weighted completion time scheduling with ready time and learning effect”, International Journal of Advanced Manufacturing Technology, doi: 10.1007/s00170-010-3022-7, 2010.
- [46] Wu, C.C. and Lee, W.C., “A note on the total completion time problem in a permutation flowshop with a learning effect”, European Journal of Operational Research, 192, 343-347, 2009.
- [47] Wu, C.C., Lee, W.C., and Wang, W.C., “A two-machine flowshop maximum tardiness scheduling problem with a learning effect”, International Journal of Advanced Manufacturing Technology, 31, 743-750, 2007.
- [48] Yelle, L.E., “The learning curve: historical review and comprehensive survey”, Decision Sciences, 10, 302-328, 1979.
- [49] Yin, Y., Xu, D., Sun, K., and Li, H., “Some scheduling problems with general position-dependent and time-dependent learning effects”, Information Sciences, 179, 2416-2425, 2009.

- [50] Zhang Y., and Li, X., “Estimation of distribution algorithm for permutation flow shops with total flowtime minimization”, Computers & Industrial Engineering, 60, 706-718, 2011.
- [51] Zhu, Z., Sun, L., Chu, F., and Liu, M., “Single-machine group scheduling with resource allocation and learning effect”, Computers & Industrial Engineering, 60, 148-157, 2011.

