Computers & Industrial Engineering 63 (2012) 302–312

Contents lists available at SciVerse ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

Bi-criteria minimization for the permutation flowshop scheduling problem with machine-based learning effects ☆

Yu-Hsiang Chung *, Lee-Ing Tong

Department of Industrial Engineering and Management, National Chiao Tung University, 1001 University Road, Hsinchu 300, Taiwan, ROC

ARTICLE INFO

Article history:
Received 28 September 2011
Received in revised form 13 March 2012
Accepted 14 March 2012
Available online 23 March 2012

Keywords:
Scheduling
Learning effect
Flowshop
Total completion time
Makespan

ABSTRACT

In traditional scheduling problems, the processing time for the given job is assumed to be a constant regardless of whether the job is scheduled earlier or later. However, the phenomenon named "learning effect" has extensively been studied recently, in which job processing times decline as workers gain more experience. This paper discusses a bi-criteria scheduling problem in an m-machine permutation flowshop environment with varied learning effects on different machines. The objective of this paper is to minimize the weighted sum of the total completion time and the makespan. A dominance criterion and a lower bound are proposed to accelerate the branch-and-bound algorithm for deriving the optimal solution. In addition, the near-optimal solutions are derived by adapting two well-known heuristic algorithms. The computational experiments reveal that the proposed branch-and-bound algorithm can effectively deal with problems with up to 16 jobs, and the proposed heuristic algorithms can yield accurate near-optimal solutions.

1. Introduction

In classical scheduling problems, all the processing times of jobs are often assumed to be fixed and known before processing the jobs (Pinedo, 2002). However, in reality, the workers acquire experience after repetitiously operating similar tasks in many practical environments. As a result, the processing times of the jobs decline as the skill of the workers improves. This phenomenon is known as the "learning effect".

Biskup (1999) addressed a learning effect model in a single-machine scheduling problem in which the processing time of the job is a function of its position in a schedule. The problems for minimizing the deviation from a common due date and minimizing the total flow time were demonstrated to be polynomially solvable. Subsequently, the learning effect has received extensive discussion in the scheduling field (Janiak & Rudek, 2008, 2010; Mosheiov & Sidney, 2003). Biskup (2008) proposed a detailed review of scheduling problems with learning effect in which the existing models are distinguished into two distinct types: the position-based learning and the sum-of-processing-time-based learning types. The position-based learning effect is affected by the number of processed jobs. Meanwhile, the sum-of-processing-time-based learning effect is influenced by the total processing times of the finished jobs.

With regard to the position-based learning effect, Lee, Wu, and Hsu (2010) 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. Toksari (2011) 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. In addition, Zhu, Sun, Chu, and Liu (2011) 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 total weighted sum of the makespan and the total resources constrained remain polynomial solvable. Huang, Wang, and Wang (2011) 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. Moreover, Lee and Lai (2011) 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.

In terms of the sum-of-processing-time-based learning effect, Koulamas and Kyparisis (2007) indicated that employees learn

☆ This manuscript was processed by Area Editor (T.C. Edwin Cheng).

* Corresponding author. Tel.: +886 928 395863.
E-mail address: yhchung.iem96g@nctu.edu.tw (Y.-H. Chung).

0360-8352/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cie.2012.03.009

more when performing jobs with a longer processing time. They brought in a sum-of-job-processing-time-based learning effect scheduling model and showed that the makespan and the total completion time problems for the single machine setting, and two-machine flowshop setting with ordered job processing times remain polynomially solvable. Furthermore, Cheng, Lai, Wu, and Lee (2009) introduced a learning effect model into a single-machine scheduling problem. The actual job processing time is derived from the sum of the logarithm of the processing times of processed jobs. They showed that the makespan and total completion time problems are polynomially solvable. Wang and Wang (2011) 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. In addition, Cheng, Cheng, Wu, Hsu, and Wu (2011) 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.

The position-based and the sum-of-processing-time-based learning effects have been concurrently considered in recent literatures. Cheng, Wu, and Lee (2008) developed a model of the learning effect on a single machine in which the actual processing time of the job is a function of the total normal processing time of processed jobs and the position of the job in a schedule. They then proved that the makespan and total completion time problems remain polynomially solvable. Lai and Lee (2011) 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. Furthermore, Yin, Xu, Sun, and Li (2009) considered learning effect in some single-machine and m-machine flowhop scheduling problems. The actual job processing time is a function of the total normal processing times of the jobs already processed and the number of processed jobs. Lee and Wu (2009) developed a general learning model that associates with the position-based and sum-of-processing-time-based learning effects. They then showed that the single-machine makespan and the total completion time problems are polynomially solvable, and proposed polynomial-time optimal solutions to minimize the makespan and total completion time under certain agreeable conditions for the flowshop setting.

In terms of the flowshop scheduling problems, Johnson (1954) was the pioneer for discussing this topic. Chung, Flynn, and Kirca (2002) investigated an m-machine flowshop problem with a total completion time objective. Then a brand-and-bound algorithm incorporated with a creative lower bound and a dominance rule was conducted to derive the optimal solution. Chung, Flynn, and Kirca (2006) studied a total tardiness minimization scheduling problem in an m-machine flowshop environment. They sought the optimal solution by implementing a branch-and-bound algorithm. In addition, the learning effect has been recently introduced into flowshop scheduling problems (Lee and Wu (2004), Wu, Lee, and Wang (2007)). Chen, Wu, and Lee (2006) considered a bi-criteria two-machine flowshop scheduling problem with the learning effect in which the objective is to minimize the weighted sum of the total completion time and the maximum tardiness. They established a branch-and-bound algorithm and two heuristic algorithms to obtain the optimal and near-optimal solutions. Li, Hsu, Wu, and Cheng (2011) 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. Additionally, Wang and Xia (2005) considered flowshop scheduling problems with a learning effect. They demonstrated examples to prove that the Johnson's rule is not the optimal method to minimize the makespan problem for a two-machine setting with learning consideration. Then they showed that two special cases remained polynomially solvable with makespan and total completion time objectives. Wu and Lee (2009) discussed a flowshop scheduling problem with a learning effect for minimizing the total completion time and then utilized a branch-and-bound algorithm to obtain the optimal solution. Then they adapted four well-known heuristic algorithms to derive the near-optimal solutions and compare the proposed heuristic algorithms.

Due to the complexity of the flowshop environment, most of the researchers have devoted to discovering near-optimal solutions. Nawaz, Enscore, and Ham (1983) investigated an m-machine flowshop scheduling problem with the makespan objective. They stated that jobs with larger total processing times have higher priority and should be scheduled earlier. Then they showed that the algorithm they proposed has remarkable performance, particularly in the large job-sized problems. Subsequently, Liu and Ong (2002) and Ruiz and Maroto (2005) pointed out that the algorithm Nawaz et al. (1983) proposed is preferable to other existing polynomial algorithms for the m-machine flowshop scheduling problem with makespan objective. Framinan, Gupta, and Leisten (2004) presented a review and classification for the heuristic algorithms with a makespan objective. Furthermore, Framinan and Leisten (2003) considered an m-machine flowshop scheduling problem for minimizing the mean flow time. They established an efficient constructive heuristic algorithm by first utilizing the concept of the algorithm proposed by Nawaz et al. (1983), and then implemented a general pairwise interchange movement to improve the solutions. Thereafter, Wu and Lee (2009) indicated that the heuristic algorithm proposed by Framinan and Leisten (2003) is recommended to approximate the total completion time for the flowshop scheduling problem with a general position-based learning effect. In addition, Wang, Pan, and Tasgetiren (2011) 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. Zhang and Li (2011) addressed an estimation of distribution algorithm for a permutation flowshop scheduling problem with the objective of minimizing the total flowtime.

In most of the literature for the flowshop scheduling problems with learning effects, it is assumed that the learning effects are identical on all machines. Therefore, in this paper, the machine-based learning effect is introduced to the model of Biskup (1999) for the m-machine permutation flowshop setting, in which the learning effects are varied on different machines. In scheduling field, most of the objective functions are relative to the completion time of the tasks, and proposing approaches to minimize the completion time is an important topic in many studies. Therefore, two widely used objective functions are considered simultaneously in this paper, which are total completion time and makespan. Due to the combinative performance of these two objective functions is emphasized, the objective function of proposed problem in this paper is to minimize the weighted sum of total completion time and makespan. The outline of this paper is constructing algorithms for obtaining the optimal and near-optimal solutions of proposed problem, and utilizing the computational experiment to evaluate the performance of proposed algorithms. The description of the remaining sections of this paper is structured as follows. The formulation of the problem is elaborated on Section 2. Then four heuristic algorithms are proposed in Section 3 to obtain the near-optimal solution. A dominance criterion and a lower bound of the branch-and-bound algorithm are established in Section 4 for

optimizing the proposed problem. The computational experiments are implemented in Section 5 to assess the performances of all proposed algorithms. Eventually, the conclusions are represented in Section 6.

2. Problem description

The following notations are utilized throughout this paper.

N: Set of jobs which contains n jobs, i.e., $N = \{1, 2, \ldots, n\}$.
S: Subset of N with s scheduled jobs.
U: Subset of N with $n - s$ unscheduled jobs.
m: Number of machines.
M_i: ith machine, where $i = 1, 2, \ldots, m$.
J_j: Job j, where $j = 1, 2, \ldots, n$.
p_{ij}: Basic 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 $a_i < 0$ for $i = 1, 2, \ldots, m$.
$[]$: The symbol which denotes the job order in a sequence.
α: The weight of the objective function with $0 \leqslant \alpha \leqslant 1$.
$C_{i[r]}(\theta)$: The completion time at the position r on M_i in sequence θ.
LB: The lower bound for the objective based on the given node.

The details of the proposed problem with the machine-based learning effect in an m-machine permutation flowshop environment are 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 denote as $O_{i,j}$ for $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$. For the processing procedure, the starting time of $O_{i,j}$ must be the larger one of the completion times of $O_{i-1,j}$ and $O_{i,j-1}$. In addition, a permutation flowshop does not allow sequence changes between machines (Pinedo, 2002). Therefore, the sequence of jobs is identical to all the machines in this paper. Since the machine-based learning effect is considered, the actual processing time p_{ijr} of J_j scheduled at position r on M_i declines based on its position, i.e.,

$$p_{ijr} = p_{ij} r^{a_i}$$

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

The aim of this paper is to seek a sequence for minimizing the weighted sum of total completion time and makespan. And then the proposed problem is defined as $F_m|p_{ijr} = p_{ij} r^{a_i}|\alpha \sum_{j=1}^{n} C_j + (1-\alpha) C_{\max}$.

3. Heuristic algorithms

For the m-machine permutation flowshop scheduling problem, the total completion time problem is demonstrated to be a strongly NP-hard problem for $m \geqslant 2$ without considering the learning effect (Lenstra, Rinnooy Kan, & Brucker, 1977). In addition, Garey, Johnson, and Sethi (1976) showed that the classical makespan problem is NP-hard for $m \geqslant 3$. Therefore, the proposed problem in this paper is conjectured to be a NP-hard problem. 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 developing efficient heuristic algorithms to derive the near-optimal solution. In this paper, four heuristic algorithms are proposed and denoted as *NEH*, *FL*, *NEH_W* and *FL_W*. Since the objective function in this paper consists of the makespan and the total completion time, *NEH* and *FL* are respectively adapted from the heuristic algorithm proposed in Nawaz et al. (1983) and Framinan and Leisten (2003) by considering the learning effect and adjusting the objective function to the bi-criteria one proposed in this paper. In the procedure of proposed heuristic algorithms, the jobs with larger total processing time (i.e. $\sum_{i=1}^{m} p_{ij}$ for $j = 1, 2, \ldots, n$) have higher priority to be selected in *NEH*, while smaller in *FL*. In addition, since the machine-based learning effects are considered in this paper, 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 total weighted processing time (i.e. $\sum_{i=1}^{m} w_i p_{ij}$ for $j = 1, 2, \ldots, n$) to determine the priority of the jobs, in which the machines with weaker learning effect have larger weight. Eventually, the procedures of *NEH_W* and *FL_W* are detailed as follows.

3.1. NEH_W algorithm

Step 1: Set sequence *PS* and *US* with empty set.
Step 2: Arrange the jobs in descending order of the total weighted normal processing time, and then 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 $k \leqslant n$, go to Step 4. Otherwise, the near-optimal sequence is set as *PS*.

3.2. FL_W algorithm

Step 1: Set sequence *PS* and *US* with empty set.
Step 2: Arrange the jobs in ascending order of the total weighted normal processing time, and then 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 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 $k \leqslant n$, go to Step 4. Otherwise, the near-optimal sequence is set as *PS*.

4. Branch-and-bound algorithm

In order to seek the optimal solution, the branch-and-bound algorithm is implemented in this paper. For the proposed branch-and-bound algorithm, we addressed a dominance criterion modified from Chung et al. (2002) and a lower bound incorporated with the Hungarian method, to facilitate the procedure for deriving the optimal solution. In this section, the specification of the dominance criterion and the lower bound are demonstrated. Eventually, the summary of the proposed branch-and-bound algorithm is represented.

4.1. Theorem and corollary of the dominance criterion

A rule is represented in following theorem which determines the dominance from two varied sequences concluding the same jobs set. If a sequence is dominated by another one, the node based on the sequence is eliminated in the branching tree.

Theorem 4.1. *There are two sequence of set N, that is $\theta_1 = (\sigma_1, \pi)$ and $\theta_2 = (\sigma_2, \pi)$, in which σ_1 and σ_2 denote two different sequence of set S, and π denotes a sequence of set U. If*

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

then σ_1 dominates σ_2.

Proof. For $k = 1, 2, \ldots, m$, we have

$$C_{k[n]}(\theta_1)=\max_{1\leqslant i\leqslant k}\{C_{i[n-1]}(\theta_1)+\sum_{u=i}^{k}p_{u[n]}n^{a_u}\}\equiv C_{i_1[n-1]}(\theta_1)+\sum_{u=i_1}^{k}p_{u[n]}n^{a_u} \tag{2a}$$

for some $1 \leqslant i_1 \leqslant k$

Similarly,

$$C_{k[n]}(\theta_2)\equiv C_{i_2[n-1]}(\theta_2)+\sum_{u=i_2}^{k}p_{u[n]}n^{a_u} \tag{2b}$$

for some $1 \leqslant i_2 \leqslant k$ (2b).

From Eqs. (2a) and (2b), we have

$$\begin{aligned}C_{k[n]}(\theta_1)-C_{k[n]}(\theta_2)&=\left[C_{i_1[n-1]}(\theta_1)+\sum_{u=i_1}^{k}p_{u[n]}n^{a_u}\right]\\&\quad-\left[C_{i_2,[n-1]}(\theta_2)+\sum_{u=i_2}^{k}p_{u[n]}n^{a_u}\right]\\&\leqslant\left[C_{i_1[n-1]}(\theta_1)+\sum_{u=i_1}^{k}p_{u[n]}n^{a_u}\right]\\&\quad-\left[C_{i_1,[n-1]}(\theta_2)+\sum_{u=i_1}^{k}p_{u[n]}n^{a_u}\right]\end{aligned}$$

$$\begin{aligned}\text{Then } C_{k[n-1]}(\theta_1)-C_{k[n-1]}(\theta_2)&\leqslant C_{i_1[n-1]}(\theta_1)-C_{i_1[n-1]}(\theta_2)\\&\leqslant\max_{1\leqslant i\leqslant m}\{C_{i[n-1]}(\theta_1)-C_{i[n-1]}(\theta_2)\}\end{aligned}$$

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

$$C_{m[s+l]}(\theta_1)-C_{m[s+l]}(\theta_2)\leqslant\max_{1\leqslant i\leqslant m}\{C_{i,[s]}(\sigma_1)-C_{i,[s]}(\sigma_2)\} \tag{3}$$

where $1 \leqslant l \leqslant n - s$.

From Eq. (3), we have

$$\begin{aligned}&[\alpha\sum_{j=1}^{n}C_{m[j]}(\theta_1)+(1-\alpha)C_{m[n]}(\theta_1)]-[\alpha\sum_{j=1}^{n}C_{m[j]}(\theta_2)+(1-\alpha)C_{m[n]}(\theta_2)]\\&\qquad\leqslant\alpha\sum_{j=1}^{s}[C_{m[j]}(\sigma_1)-C_{m[j]}(\sigma_2)]+[\alpha(n-s-1)\\&\qquad+1]\max_{1\leqslant i\leqslant m}\{C_{i[s]}(\sigma_1)-C_{i[s]}(\sigma_2)\}\end{aligned} \tag{4}$$

The value for the left side of Eq. (4) is negative by Eq. (1)and it implies θ_1 dominates θ_2.

Therefore, we have σ_1 dominates σ_2. □

In this paper, the theorem is simplified as the corollary which requires considering two adjacent jobs. The corollary is utilized in the proposed branch-and-bound algorithm and presented below.

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

$$\begin{aligned}&\alpha[C_{mJ_2}(S_2)+C_{mJ_1}(S_2)-C_{mJ_1}(S_1)-C_{mJ_2}(S_1)]\\&\quad>[\alpha(n-s-1)+1]\max_{1\leqslant i\leqslant m}\{C_{ij_2}(S_1)-C_{ij_1}(S_2)\}\end{aligned} \tag{5}$$

then S_1 dominates S_2.

4.2. Calculation of the lower bound

In addition to the dominance criterion, another procedure to eliminate nodes in the branching tree is calculating the lower bound of the objective value. In this paper, we establish a lower bound to speed up the procedure of the proposed branch-and-bound algorithm. The proposed lower bound requires $O(mn^3)$ computation time, and is presented as follows.

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

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

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

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

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

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

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

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

By an induction, the underestimated value of the completion time for $(s + l)$th job on M_m based on M_k is presented as

$$C_{k[s]}(\theta)+\sum_{v=1}^{l}p_{k[s+v]}(s+v)^{a_k}+\sum_{i=k+1}^{m}p_{i[s+l]}(s+l)^{a_i} \tag{6}$$

Then we have

$$\begin{aligned}\alpha\sum_{j=1}^{n}C_{m[j]}(\theta)+(1-\alpha)C_{m[n]}(\theta)&\geqslant\alpha\sum_{j=1}^{s}C_{m[j]}(\theta)+[\alpha(n-s-1)+1]C_{k[s]}(\theta)\\&\quad+\sum_{l=1}^{n-s}[\alpha(n-s-l)+1](s+l)^{a_k}p_{k[s+l]}\\&\quad+\sum_{l=1}^{n-s}[(\alpha+I(l))\sum_{i=k+1}^{m}p_{i[s+l]}(s+l)^{a_i}]\end{aligned}$$

where

$$I(l)=\begin{cases}1-\alpha, & l=n-s\\0, & l\neq n-s\end{cases} \tag{7}$$

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

$$\begin{aligned}\alpha\sum_{j=1}^{n}C_{m[j]}(\theta)+(1-\alpha)C_{m[n]}(\theta)&\geqslant\alpha\sum_{j=1}^{s}C_{m[j]}(\theta)+[\alpha(n-s-1)+1]C_{k[s]}(\theta)\\&\quad+\sum_{l=1}^{n-s}[\alpha(n-s-l)+1](s+l)^{a_k}p_{k(s+l)}\\&\quad+\sum_{l=1}^{n-s}[(\alpha+I(l))\sum_{i=k+1}^{m}p_{i[s+l]}(s+l)^{a_i}]\end{aligned} \tag{8}$$

where $p_{k(s+l)}$ denotes the lth smallest basic processing time on M_k of the job in set U, and it implies that $p_{k(s+1)} \leqslant p_{k(s+2)} \leqslant \cdots \leqslant p_{k(n)}$

Table 1
The index set of the learning effects.

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

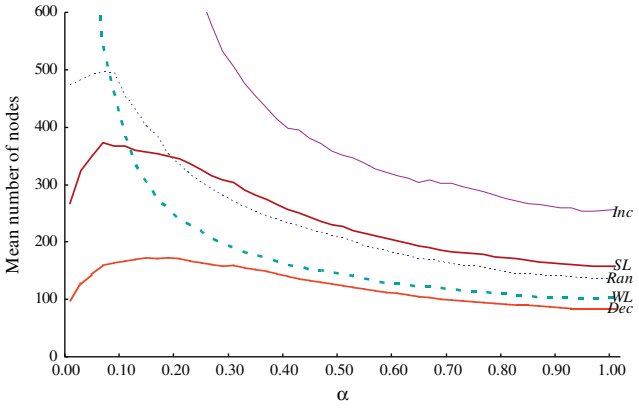

Fig. 1. The number of nodes for the branch-and-bound algorithm under different α ($n = 10$).

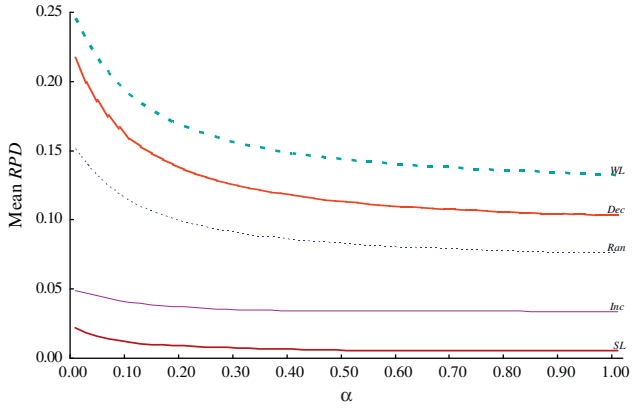

Fig. 2. The relative deviation percentage of the learning patterns for the optimal objective value under different α ($n = 10$).

Table 2
The performance of the branch-and-bound algorithm.

n	m	Pattern	$\alpha = 0.25$				$\alpha = 0.50$				$\alpha = 0.75$			
			Number of nodes		Cpu times		Number of nodes		Cpu times		m		Cpu times	
			Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max
12	5	*Ran*	1550.43	9213	1.84	8.34	1145.36	8605	1.55	8.63	901.69	6087	1.36	7.14
		Inc	3992.39	34418	2.66	19.56	2548.42	14103	1.98	10.08	1893.09	10818	1.65	7.86
		Dec	429.49	3109	0.40	2.09	347.98	3010	0.37	1.98	297.70	2682	0.34	1.78
		SL	1561.97	10346	1.01	4.20	1170.21	10062	0.88	4.02	969.24	7022	0.79	3.30
		WL	1045.37	11976	0.86	4.97	755.21	7559	0.73	4.58	612.87	6531	0.66	4.09
	7	*Ran*	2328.46	19105	1.32	9.09	1624.33	22124	1.08	10.16	1049.45	12119	0.88	5.38
		Inc	6377.07	54718	5.68	25.11	3816.40	25625	4.19	20.97	2713.35	18196	3.45	17.97
		Dec	635.63	6673	0.89	6.38	486.78	5711	0.80	5.70	363.18	3977	0.71	5.39
		SL	1920.45	10152	2.03	8.16	1278.22	6932	1.67	7.67	983.96	7241	1.45	7.92
		WL	1277.90	10061	1.70	10.41	871.76	6145	1.38	7.09	709.46	5993	1.22	7.05
14	5	*Ran*	8397.72	97712	13.43	99.28	5727.28	56055	10.91	74.48	4845.95	46274	9.82	73.30
		Inc	31883.81	418776	44.54	372.50	17452.31	169405	30.27	216.80	13144.88	97435	25.12	175.41
		Dec	1769.26	12080	3.22	16.92	1374.09	9108	2.94	17.28	1179.42	10186	2.76	17.73
		SL	10382.89	115578	12.64	55.16	6689.93	49969	10.17	44.33	5140.81	28730	8.80	43.48
		WL	3992.79	38389	10.23	84.28	2948.51	28213	8.34	67.02	2563.70	25591	7.50	62.02
	7	*Ran*	16271.02	142636	32.37	290.52	9917.81	122468	24.74	274.36	7584.29	99124	21.15	244.52
		Inc	74506.94	1980840	102.24	981.52	32558.74	322750	67.46	399.08	20916.03	124587	52.36	355.20
		Dec	3317.76	87609	7.91	148.41	2486.66	74205	6.95	139.94	1767.69	47392	5.95	108.64
		SL	12229.56	86179	23.97	123.91	8137.72	42532	19.40	99.19	6565.91	35383	17.09	89.53
		WL	8076.68	158484	22.38	389.20	5646.80	105127	17.57	280.86	4478.99	76297	15.16	224.95
16	5	*Ran*	105044.01	2408452	349.62	4578.30	55620.78	600087	246.60	2274.17	39622.63	426218	201.16	1909.00
		Inc	201758.16	2061604	514.57	3209.59	120521.06	1270538	350.70	2457.27	93087.06	1411099	288.07	2618.19
		Dec	7520.64	107243	24.58	200.00	5155.33	41868	21.69	156.70	4148.91	37436	19.70	140.19
		SL	66831.54	626475	233.56	1304.53	43107.35	360252	186.24	931.81	35647.61	289344	166.05	888.14
		WL	31332.85	957031	103.05	1421.47	20896.89	575854	78.99	992.58	16706.67	402084	68.83	781.89
	7	*Ran*	116674.80	2455027	238.17	3462.94	69780.49	1029782	167.75	2050.94	48757.11	674012	201.16	1909.00
		Inc	740132.88	5188924	2252.10	16791.42	383647.16	4065895	1369.95	13054.45	263149.47	3726265	1034.44	11907.66
		Dec	18756.10	145112	83.65	545.95	13486.31	101862	73.19	504.02	11616.93	89494	67.83	524.33
		SL	105188.86	1447154	185.16	1820.66	62497.38	842215	130.71	864.78	44370.55	531122	108.13	666.53
		WL	55492.39	1245433	287.12	6346.81	36222.31	1007826	218.60	5101.81	29516.95	783194	188.49	4129.69

In order to underestimate the final term on the right side of Eq. (8), a Hungarian method is applied and the matrix of which is formed as follows.

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

where $p_{ij_{s+l}}$ is the basic processing time on M_i of J_{s+l} in set U for $1 \leqslant l \leqslant n-s$. In the matrix, the information of a given job only can be assigned to a position from position $s+1$ to n. And then the information of all jobs are sum up as H_k which denotes the optimal value of the proposed Hungarian method. Hence, the underestimated value of the objective function based on M_k for θ is presented as

$$\alpha\sum_{j=1}^{s} C_{m[j]}(\theta) + [\alpha(n-s-1)+1]C_{k[s]}(\theta) + \sum_{l=1}^{n-s}[\alpha(n-s-l)+1](s+l)^{a_k} p_{k(s+l)} + H_k \quad (9)$$

In order to make the lower bound stricter, the lower bound is evaluated as

$$LB = \alpha\sum_{j=1}^{s} C_{m[j]}(\theta) + \max_{1\leqslant k\leqslant m}\{[\alpha(n-s-1)+1]C_{k[s]}(\theta) + \sum_{l=1}^{n-s}[\alpha(n-s-l)+1](s+l)^{a_k} p_{k(s+l)} + H_k\} \quad (10)$$

4.3. Summary of the branch-and-bound algorithm

In this paper, the depth-first search plus forward manner is implemented in the branching procedure, i.e. the nodes are spread from $(1,-,\cdots,-)$ to $(1,2,-,\cdots,-)$, and finally to $(n,n-1,\cdots,1)$. The advantages of the depth-first search are storing less numbers of dynamic nodes and seeking the bottom node rapidly to derive a feasible solution. The summary is listed as follows.

Step 1: {Initialization} Implement the heuristic algorithms to obtain an initial incumbent solution.
Step 2: {Branching} Utilize Corollary 4.1 to eliminate nodes.
Step 3: {Bounding} If the lower bound exceeds the incumbent solution, eliminate the node and its offspring.

5. Computational results

Several computational experiments are implemented in this section 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 3.25 GB RAM with Windows XP. Since the machine-based learning effect is considered, the issue for allocating the learning effects to the machines is discussed in this paper. In general, the stronger learning effect is assigned to the machine with heavier workload, like the concept of bottlenecks. In order to verify this viewpoint, five learning patterns under the same learning indices set are proposed to discuss the influence on the proposed algorithms. The five learning patterns are denoted as *Ran, Inc, Dec, SL* and *WL* 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 $i = 1, 2, \ldots, m$.
WL: The weaker learning effects are assigned to the machines with the larger value of $\sum_{j=1}^{n} p_{ij}$ for $i = 1, 2, \ldots, m$.

For all computational experiments in this paper, the basic processing times are randomly generated from a discrete uniform distribution over the integer 1–100 (i.e., $p_{ij} \sim U(1,100)$ for $j = 1, 2, \ldots, n$ and $i = 1, 2, \ldots, m$). The sets for the learning indices under different number of machines (i.e., a_i for $i = 1, 2, \ldots, m$) are shown in Table 1. And then the simulated results output the number of nodes, the execution time and the objective value to evaluate the performance of proposed algorithm under different experimental parameters.

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 respectively set as 10 and 5 and 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., $\alpha = 0, 0.02, 0.04, \ldots, 1$. The five learning patterns and 51 different α are considered in each example and the results are illustrated in Figs. 1 and 2.

In Fig. 1, the mean numbers of nodes for all experimental conditions are illustrated. It is observed that the problem proposed in this paper is easier to solve as α increases with respect to the trend of the mean number of nodes. The reason is that the corollary and the lower bound are more efficient in the branch-and-bound

Table 3
The comparison among five learning patterns for the optimal objective value.

n	m	Patten	RDP_O					
			$\alpha = 0.25$		$\alpha = 0.50$		n	
			Mean	Max	Mean	Max	Mean	Max
12	5	*Ran*	1.089	1.297	1.078	1.275	1.073	1.267
		Inc	1.049	1.292	1.046	1.275	1.044	1.267
		Dec	1.125	1.356	1.109	1.324	1.103	1.310
		SL	1.007	1.046	1.006	1.041	1.005	1.039
		WL	1.161	1.346	1.144	1.316	1.137	1.303
	7	*Ran*	1.070	1.195	1.063	1.167	1.060	1.163
		Inc	1.025	1.119	1.024	1.105	1.024	1.103
		Dec	1.125	1.237	1.109	1.217	1.103	1.212
		SL	1.007	1.064	1.006	1.057	1.005	1.057
		WL	1.136	1.301	1.121	1.273	1.115	1.264
14	5	*Ran*	1.099	1.392	1.090	1.346	1.086	1.327
		Inc	1.054	1.200	1.050	1.197	1.048	1.194
		Dec	1.132	1.392	1.117	1.346	1.111	1.327
		SL	1.006	1.067	1.005	1.046	1.005	1.037
		WL	1.171	1.398	1.155	1.351	1.148	1.332
	7	*Ran*	1.083	1.327	1.075	1.296	1.071	1.288
		Inc	1.022	1.157	1.022	1.136	1.021	1.126
		Dec	1.141	1.330	1.125	1.299	1.119	1.288
		SL	1.006	1.056	1.005	1.039	1.004	1.038
		WL	1.157	1.328	1.141	1.295	1.135	1.282
16	5	*Ran*	1.095	1.271	1.086	1.247	1.083	1.239
		Inc	1.068	1.301	1.063	1.286	1.061	1.279
		Dec	1.136	1.413	1.122	1.376	1.117	1.365
		SL	1.006	1.060	1.005	1.054	1.005	1.052
		WL	1.180	1.416	1.166	1.379	1.160	1.369
	7	*Ran*	1.084	1.276	1.075	1.250	1.072	1.242
		Inc	1.031	1.183	1.029	1.167	1.028	1.158
		Dec	1.128	1.411	1.116	1.371	1.111	1.357
		SL	1.008	1.068	1.007	1.061	1.007	1.061
		WL	1.158	1.405	1.143	1.364	1.137	1.349

Table 4
The performance of the heuristic algorithms ($\alpha = 0.25$).

n	m	Patten	Error percentages											
			NEH		*NEH_W*		min{*NEH*,*NEH_W*}		*FL*		*FL_W*		min{*FL*,*FLW*}	
			Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max
12	5	*Ran*	0.051	0.132	0.059	0.134	0.045	0.110	0.011	0.067	0.012	0.067	0.008	0.067
		Inc	0.033	0.097	0.037	0.105	0.029	0.097	0.010	0.056	0.012	0.054	0.007	0.045
		Dec	0.062	0.144	0.065	0.201	0.052	0.144	0.010	0.043	0.009	0.060	0.006	0.030
		SL	0.055	0.138	0.057	0.138	0.046	0.126	0.015	0.060	0.017	0.049	0.011	0.049
		WL	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	*Ran*	0.046	0.141	0.052	0.150	0.040	0.097	0.015	0.054	0.014	0.058	0.010	0.054
		Inc	0.042	0.102	0.043	0.113	0.035	0.085	0.015	0.064	0.015	0.060	0.011	0.058
		Dec	0.057	0.152	0.056	0.131	0.045	0.102	0.010	0.059	0.011	0.072	0.006	0.037
		SL	0.051	0.140	0.051	0.115	0.041	0.110	0.014	0.049	0.016	0.066	0.011	0.039
		WL	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	*Ran*	0.050	0.110	0.058	0.136	0.045	0.095	0.012	0.044	0.011	0.065	0.008	0.040
		Inc	0.034	0.093	0.039	0.108	0.030	0.093	0.010	0.044	0.011	0.052	0.007	0.035
		Dec	0.069	0.147	0.075	0.146	0.060	0.121	0.011	0.046	0.011	0.056	0.007	0.043
		SL	0.058	0.152	0.063	0.152	0.049	0.120	0.016	0.057	0.019	0.065	0.011	0.037
		WL	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	*Ran*	0.051	0.101	0.054	0.178	0.042	0.101	0.014	0.049	0.014	0.084	0.010	0.037
		Inc	0.041	0.101	0.047	0.097	0.035	0.082	0.016	0.056	0.018	0.070	0.012	0.046
		Dec	0.063	0.154	0.065	0.169	0.053	0.132	0.012	0.044	0.012	0.062	0.008	0.037
		SL	0.057	0.140	0.057	0.131	0.049	0.131	0.017	0.063	0.020	0.071	0.013	0.046
		WL	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	*Ran*	0.061	0.182	0.069	0.164	0.054	0.122	0.013	0.069	0.014	0.069	0.010	0.069
		Inc	0.041	0.147	0.043	0.155	0.035	0.079	0.011	0.049	0.011	0.061	0.008	0.044
		Dec	0.079	0.177	0.084	0.169	0.069	0.167	0.011	0.039	0.011	0.045	0.008	0.035
		SL	0.069	0.163	0.074	0.162	0.059	0.129	0.019	0.087	0.021	0.066	0.014	0.049
		WL	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	*Ran*	0.058	0.147	0.061	0.133	0.049	0.127	0.014	0.060	0.016	0.061	0.011	0.060
		Inc	0.040	0.109	0.049	0.103	0.037	0.089	0.017	0.069	0.016	0.062	0.012	0.062
		Dec	0.069	0.159	0.068	0.139	0.057	0.134	0.014	0.052	0.014	0.041	0.010	0.041
		SL	0.066	0.181	0.064	0.168	0.054	0.132	0.020	0.068	0.024	0.069	0.015	0.048
		WL	0.055	0.127	0.062	0.150	0.049	0.108	0.012	0.046	0.012	0.061	0.009	0.046

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 deviation percentage for five learning patterns is denoted as RDP_O and its mean is illustrated in Fig. 2. For each example, the RDP_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 $\lambda_{\min}$ 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 RDP_O are larger than zero.

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 and 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 2–6.

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 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 paper is NP-hard. The optimal solution is easier to be derived 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 RDP_O are recorded for all computational conditions in Table 3.

As shown in Table 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.

For the proposed heuristic algorithms, the mean and maximum error percentages under different α are reported in Tables 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 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*.

As shown in Tables 4–6, it is observed that all heuristic algorithms proposed in this paper are quite accurate since the error percentages are all less than 1%. For evaluating the influence on the performance of the heuristic algorithms, several two-way anal-

Table 5
The performance of the heuristic algorithms ($\alpha = 0.50$).

n	m	Patten	Error percentages											
			NEH		*NEH_W*		Min{*NEH*,*NEH_W*}		*FL*		*FL_W*		Min{*FL*,*FLW*}	
			Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max
12	5	*Ran*	0.056	0.153	0.063	0.146	0.049	0.128	0.010	0.065	0.012	0.064	0.008	0.064
		Inc	0.035	0.090	0.041	0.118	0.031	0.090	0.011	0.047	0.011	0.049	0.008	0.043
		Dec	0.066	0.173	0.069	0.167	0.056	0.167	0.009	0.051	0.009	0.049	0.006	0.041
		SL	0.049	0.142	0.054	0.138	0.042	0.138	0.013	0.049	0.014	0.048	0.010	0.048
		WL	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	*Ran*	0.050	0.137	0.051	0.153	0.041	0.106	0.015	0.060	0.013	0.063	0.010	0.046
		Inc	0.042	0.106	0.042	0.111	0.035	0.085	0.014	0.053	0.013	0.051	0.010	0.049
		Dec	0.063	0.181	0.057	0.136	0.048	0.118	0.010	0.050	0.010	0.050	0.007	0.050
		SL	0.052	0.137	0.051	0.130	0.042	0.125	0.013	0.051	0.015	0.049	0.010	0.037
		WL	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	*Ran*	0.056	0.124	0.062	0.161	0.048	0.104	0.013	0.056	0.012	0.056	0.008	0.047
		Inc	0.038	0.093	0.043	0.137	0.033	0.086	0.012	0.045	0.011	0.045	0.009	0.045
		Dec	0.079	0.169	0.085	0.186	0.068	0.156	0.011	0.044	0.011	0.064	0.007	0.028
		SL	0.058	0.129	0.066	0.151	0.051	0.122	0.016	0.064	0.019	0.082	0.012	0.055
		WL	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	*Ran*	0.051	0.117	0.055	0.115	0.044	0.115	0.012	0.043	0.014	0.054	0.010	0.037
		Inc	0.041	0.116	0.046	0.107	0.036	0.090	0.016	0.063	0.016	0.063	0.012	0.063
		Dec	0.068	0.173	0.067	0.173	0.056	0.161	0.010	0.054	0.012	0.051	0.008	0.043
		SL	0.058	0.138	0.056	0.144	0.047	0.138	0.016	0.049	0.018	0.069	0.013	0.049
		WL	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	*Ran*	0.067	0.203	0.072	0.185	0.058	0.135	0.013	0.063	0.014	0.073	0.010	0.060
		Inc	0.047	0.139	0.047	0.161	0.039	0.093	0.012	0.072	0.013	0.051	0.009	0.051
		Dec	0.088	0.196	0.090	0.167	0.076	0.167	0.013	0.059	0.011	0.068	0.008	0.047
		SL	0.071	0.164	0.075	0.172	0.060	0.131	0.019	0.074	0.021	0.147	0.014	0.047
		WL	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	*Ran*	0.061	0.150	0.065	0.148	0.053	0.144	0.014	0.061	0.014	0.048	0.010	0.046
		Inc	0.044	0.095	0.048	0.098	0.038	0.088	0.016	0.057	0.014	0.062	0.011	0.057
		Dec	0.074	0.157	0.072	0.179	0.062	0.157	0.013	0.048	0.013	0.053	0.008	0.038
		SL	0.066	0.146	0.065	0.177	0.055	0.146	0.019	0.059	0.019	0.073	0.013	0.040
		WL	0.061	0.143	0.069	0.151	0.056	0.133	0.012	0.058	0.013	0.052	0.008	0.039

ysis of variance (ANOVA) are performed to test three hypotheses at the 0.05 level of significance. The first two null hypotheses are assumed as that the mean error percentages of the given heuristic algorithm are all identical among the three α settings, and five learning patterns, respectively. The last null hypothesis is assumed as that there is no interaction between α and learning patterns. And then the results are reported in Table 7.

As shown in Table 7, it is observed that α does not have a significant effect on the accuracy for all heuristic algorithms except *NEH*. Then it is shown in Tables 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 4–6 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 pattern 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 for selecting jobs which are 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 paper. 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 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 deviation percentage for all heuristic algorithms is denoted as RDP_H, and its mean and maximum values are listed in Table 8. For each example, the RDP_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 $\mu_{\min}$ is the minimum among all μ.

As shown in Table 8 that *FL* and *FL_W* are both better than min{*NEH*,*NEH_W*} in terms of the RDP_H. It implies that the heuristic algorithm proposed by Framinan and Leisten (2003) is more proper than the algorithm proposed by Nawaz et al. (1983) to obtain the near-optimal solution for the problem proposed in this paper. 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 paper.

6. Conclusions

In this paper, an m-machine permutation flowshop scheduling problem with machine-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

Table 6
The performance of the heuristic algorithms ($\alpha = 0.75$).

n	*m*	Patten	Error percentages											
			NEH		*NEH_W*		Min{*NEH,NEH_W*}		*FL*		*FL_W*		Min{*FL,FLW*}	
			Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max
12	5	*Ran*	0.058	0.159	0.064	0.157	0.050	0.132	0.012	0.064	0.011	0.044	0.008	0.044
		Inc	0.036	0.086	0.039	0.092	0.031	0.081	0.011	0.044	0.011	0.053	0.008	0.035
		Dec	0.069	0.149	0.070	0.170	0.057	0.122	0.009	0.052	0.009	0.044	0.006	0.030
		SL	0.056	0.179	0.057	0.144	0.047	0.144	0.013	0.046	0.016	0.055	0.010	0.041
		WL	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	*Ran*	0.050	0.137	0.052	0.133	0.042	0.109	0.013	0.051	0.013	0.046	0.009	0.043
		Inc	0.042	0.104	0.041	0.091	0.034	0.081	0.013	0.064	0.012	0.051	0.010	0.051
		Dec	0.067	0.191	0.057	0.137	0.050	0.129	0.009	0.040	0.009	0.041	0.006	0.037
		SL	0.054	0.143	0.052	0.132	0.042	0.130	0.014	0.060	0.015	0.045	0.010	0.043
		WL	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	*Ran*	0.057	0.131	0.064	0.131	0.050	0.109	0.013	0.054	0.012	0.050	0.008	0.045
		Inc	0.039	0.103	0.043	0.105	0.034	0.083	0.012	0.040	0.013	0.059	0.009	0.040
		Dec	0.081	0.180	0.082	0.199	0.068	0.159	0.011	0.048	0.011	0.053	0.007	0.044
		SL	0.060	0.137	0.066	0.154	0.050	0.127	0.017	0.084	0.018	0.057	0.012	0.043
		WL	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	*Ran*	0.051	0.126	0.056	0.152	0.044	0.126	0.013	0.047	0.013	0.050	0.010	0.042
		Inc	0.043	0.120	0.045	0.103	0.036	0.103	0.017	0.060	0.016	0.058	0.013	0.056
		Dec	0.068	0.180	0.068	0.178	0.056	0.167	0.010	0.045	0.011	0.067	0.007	0.045
		SL	0.058	0.126	0.057	0.151	0.047	0.125	0.016	0.053	0.019	0.065	0.012	0.053
		WL	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	*Ran*	0.069	0.222	0.071	0.173	0.059	0.153	0.016	0.091	0.014	0.056	0.010	0.042
		Inc	0.048	0.151	0.048	0.164	0.040	0.094	0.012	0.064	0.013	0.039	0.008	0.035
		Dec	0.089	0.207	0.090	0.179	0.075	0.149	0.011	0.043	0.011	0.068	0.008	0.032
		SL	0.072	0.169	0.074	0.172	0.060	0.144	0.020	0.076	0.022	0.160	0.014	0.076
		WL	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	*Ran*	0.063	0.157	0.065	0.155	0.054	0.151	0.014	0.046	0.015	0.055	0.010	0.044
		Inc	0.045	0.097	0.046	0.110	0.038	0.094	0.015	0.054	0.016	0.067	0.011	0.043
		Dec	0.076	0.161	0.075	0.189	0.064	0.161	0.013	0.042	0.013	0.053	0.009	0.035
		SL	0.067	0.159	0.065	0.182	0.056	0.159	0.018	0.074	0.020	0.074	0.014	0.074
		WL	0.063	0.150	0.070	0.161	0.057	0.139	0.014	0.055	0.014	0.054	0.010	0.035

Table 7
Two-way ANOVA of the error percentages for all heuristic algorithms.

Heuristic algorithm	Source	*DF*	*SS*	*MS*	*F*	*p*-Value
NEH	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			
NEH_W	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{NEH,NEH_W}	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			
FL	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			
FL_W	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{FL,FL_W}	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			

Table 8
The comparison of the heuristic algorithms for large job-sized problem ($\alpha = 0.50$).

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

dominance criterion and a lower bound is proposed to seek the optimal solution, and several heuristic algorithms are established to yield the near-optimal solutions. As shown in the computational results, the proposed problem can be dealt with up to 16 jobs within a reasonable amount of time. When the learning pattern is set as *Inc*, or if α is smaller, the proposed problem is harder to search for the optimal solution by implementing the proposed branch-and-bound algorithm. Furthermore, the performances of all proposed heuristic algorithms are accurate while min{*FL*, *FL_W*} is recommended to obtain the near-optimal solution. Finally, the issue for allocating the learning effects to the machines is discussed in this paper, and it is verified that assigning the stronger learning effects to the machines with the heavier workload might obtain the better result for the problem proposed in this paper.

References

Biskup, D. (1999). Single-machine scheduling with learning considerations. *European Journal of Operational Research, 115*, 173–178.

Biskup, D. (2008). A state-of-the-art review on scheduling with learning effect. *European Journal of Operational Research, 188*, 315–329.

Chen, P., Wu, C. C., & Lee, W. C. (2006). A bi-criteria two-machine flowshop scheduling problem with a learning effect. *Journal of the Operational Research Society, 57*, 1113–1125.

Cheng, T. C. E., Cheng, S. R., Wu, W. H., Hsu, P. H., & Wu, C. C. (2011). A two-agent single-machine scheduling problem with truncated sum-of-processing-times-based learning consideration. *Computers & Industrial Engineering, 60*, 534–541.

Cheng, T. C. E., Lai, P. J., Wu, C. C., & Lee, W. C. (2009). Single-machine scheduling with sum-of-logarithm-processing-times-based learning considerations. *Information Sciences, 179*, 3127–3135.

Cheng, T. C. E., Wu, C. C., & Lee, W. C. (2008). Some scheduling problems with sum-of-processing-times-based and job-position-based learning effects. *Information Sciences, 178*, 2476–2487.

Chung, C. S., Flynn, J., & Kirca, Ő. (2002). 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.

Chung, C. S., Flynn, J., & Kirca, Ő. (2006). A branch and bound algorithm to minimize the total tardiness for *m*-machine permutation flowshop problems. *European Journal of Operational Research, 174*, 1–10.

Framinan, J. M., Gupta, J. N. D., & Leisten, R. (2004). A review and classification of heuristics for permutation flow-shop scheduling with makespan objective. *Journal of the Operational Research Society, 55*, 1243–1255.

Framinan, J. M., & Leisten, R. (2003). An efficient constructive heuristic for flowtime minimization in permutation flow shops. *OMEGA, 31*, 311–317.

Garey, M. R., Johnson, D. S., & Sethi, R. (1976). The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research, 1*, 117–129.

Huang, X., Wang, M. Z., & Wang, J. B. (2011). Single-machine scheduling with both learning effects and deteriorating jobs. *Computers & Industrial Engineering, 60*, 750–754.

Janiak, A., & Rudek, R. (2008). A new approach to the learning effect: Beyond the learning curve restrictions. *Computers and Operations Research, 35*, 3727–3736.

Janiak, A., & Rudek, R. (2010). A note on a makespan minimization problem with a multi-ability learning effect. *OMEGA, 38*, 213–217.

Johnson, S. M. (1954). Optimal two and three-stage production schedules with setup times included. *Naval Research Logistics Quarterly, 1*, 61–67.

Koulamas, C., & Kyparisis, G. J. (2007). Single-machine and two-machine flowshop scheduling with general learning function. *European Journal of Operational Research, 178*, 402–407.

Lai, P. J., & Lee, W. C. (2011). Single-machine scheduling with general sum-of-processing-time-based and position-based learning effects. *OMEGA, 39*, 467–471.

Lee, W. C., & Lai, P. J. (2011). Scheduling problems with general effects of deterioration and learning. *Information Sciences, 181*, 1164–1170.

Lee, W. C., & Wu, C. C. (2004). Minimizing total completion time in a two-machine flowshop with a learning effect. *International Journal of Production Economics, 88*, 85–93.

Lee, W. C., & Wu, C. C. (2009). Some single-machine and m-machine flowshop scheduling problems with learning considerations. *Information Sciences, 179*, 3885–3892.

Lee, W. C., Wu, C. C., & Hsu, P. H. (2010). A single-machine learning effects scheduling problem with release times. *OMEGA, 38*, 3–11.

Lenstra, J. K., Rinnooy Kan, A. H. G., & Brucker, P. (1977). Complexity of machine scheduling problems. *Annals of Discrete Mathematics, 1*, 343–362.

Li, D. C., Hsu, P. H., Wu, C. C., & Cheng, T. C. E. (2011). Two-machine flowshop scheduling with truncated learning to minimize the total completion time. *Computers & Industrial Engineering, 61*, 655–662.

Liu, S., & Ong, H. L. (2002). A comparative study of algorithms for the flowshop scheduling problem. *Asia–Pacific Journal of Operational Research, 19*, 205–222.

Mosheiov, G., & Sidney, J. B. (2003). Scheduling with general job-dependent learning curves. *European Journal of Operational Research, 147*, 665–670.

Nawaz, M., Enscore, E. E., & Ham, I. (1983). A heuristic algorithm for the *m*-machine, *n*-job flow-shop sequencing problem. *OMEGA, 11*, 91–95.

Pinedo, M. (2002). *Scheduling: Theory, algorithms, and systems* (2nd ed.). Upper Saddle River, New Jersey: Prentice-Hall.

Ruiz, R., & Maroto, C. (2005). A comprehensive review and evaluation of permutation flowshop heuristics. *European Journal of Operational Research, 165*, 479–494.

Toksari, M. D. (2011). 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.

Wang, L., Pan, Q. K., & Tasgetiren, M. F. (2011). A hybrid harmony search algorithm for the blocking permutation flow shop scheduling problem. *Computers & Industrial Engineering, 61*, 76–83.

Wang, J. B., & Wang, J. J. (2011). Single-machine scheduling jobs with exponential learning functions. *Computers & Industrial Engineering, 60*, 755–759.
Wang, J. B., & Xia, Z. Q. (2005). Flow-shop scheduling with a learning effect. *Journal of the Operational Research Society, 56*, 1325–1330.
Wu, C. C., & Lee, W. C. (2009). A note on the total completion time problem in a permutation flowshop with a learning effect. *European Journal of Operational Research, 192*, 343–347.
Wu, C. C., Lee, W. C., & Wang, W. C. (2007). A two-machine flowshop maximum tardiness scheduling problem with a learning effect. *The International Journal of Advanced Manufacturing Technology, 31*, 743–750.
Yin, Y., Xu, D., Sun, K., & Li, H. (2009). Some scheduling problems with general position-dependent and time-dependent learning effects. *Information Sciences, 179*, 2416–2425.
Zhang, Y., & Li, X. (2011). Estimation of distribution algorithm for permutation flow shops with total flowtime minimization. *Computers & Industrial Engineering, 60*, 706–718.
Zhu, Z., Sun, L., Chu, F., & Liu, M. (2011). Single-machine group scheduling with resource allocation and learning effect. *Computers & Industrial Engineering, 60*, 148–157.