Int. J. Production Economics 141 (2013) 327–334

Contents lists available at SciVerse ScienceDirect

Int. J. Production Economics

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

Permutation flowshop scheduling to minimize the total tardiness with learning effects

Wen-Chiung Lee [a], Yu-Hsiang Chung [b,*]

[a] Department of Statistics, Feng Chia University, Taichung, Taiwan
[b] Department of Industrial & Engineering Management, National Chiao Tung University, Hsinchu 300, Taiwan

ARTICLE INFO

Article history:
Received 4 January 2011
Accepted 1 August 2012
Available online 25 August 2012

Keywords:
Scheduling
Permutation flowshop
Total tardiness
Learning effect

ABSTRACT

Scheduling with learning effects has received considerable attention recently. Often, numbers of operations have to be done on every job in many manufacturing and assembly facilities. However, it is seldom discussed in the general multiple-machine setting, especially without the assumptions of identical processing time on all the machines or dominant machines. With the current emphasis of customer service and meeting the promised delivery dates, we consider a permutation flowshop scheduling problem with learning effects where the objective is to minimize the total tardiness. A branch-and-bound algorithm and two heuristic algorithms are established to search for the optimal and near-optimal solutions. Computational experiments are also given to evaluate the performance of the algorithms.

© 2012 Elsevier B.V. All rights reserved.

1. Introduction

In classical scheduling, the job processing times are assumed to be fixed and known throughout the entire process. However, this assumption might not reflect many real-life situations. For example, Biskup (1999) pointed out that repeated processing of similar tasks improves the worker skills; workers are able to perform setup, to deal with machine operations or software, or to handle raw materials and components at a greater pace. Biskup (1999) and Cheng and Wang (2000) were among the pioneers that brought the concept of learning effects into the scheduling field. Many researchers have devoted to this young but vivid area since. Biskup (2008) provided a comprehensive review of the scheduling models and problems with learning effects.

Recently, Wang (2007) considered some single-machine problems with the effects of learning and deterioration, and proved that the makespan and the sum of completion time problems remain polynomially solvable. He also showed that the weighted shortest processing time rule and the earliest due date rule provide the optimal schedules for the weighted sum of completion time and the maximum lateness problems in some special cases. Janiak and Rudek (2008) considered a scheduling problem in which each job provides a different experience to the processor. They relaxed one of the rigorous constraints, and thus each job can provide different experience to the processor in their model. They then formulated the job processing time as a non-increasing k-stepwise function that in general is not restricted to a certain learning curve, thereby it can accurately fit every possible shape of a learning function. Lee and Wu (2004) investigated a two machine flowshop scheduling problem with learning consideration to minimize the total completion time. They utilized the branch-and-bound algorithm incorporated with several dominance properties and lower bounds to obtain the optimal solution. An accurate heuristic algorithm was also proposed to obtain the near-optimal solution. Cheng et al. (in press) studied a two-machine flowshop scheduling problem with a truncated learning function to minimize the makespan. They utilized a branch-and-bound and three heuristic algorithms to derive the optimal and near-optimal solutions. Wang (2008) studied some single-machine problems with the sum-of-processing-time-based learning effect. He showed by examples that the classical optimal rules no longer provide the optimal solutions under the proposed model. He also provided the optimal solutions for some single-machine problems under certain conditions. Cheng et al. (2008), Lee and Wu (2009), Yin et al. (2009) and Zhang and Yan (2010) considered a variety of models in which the actual job processing time not only depends on its scheduled position, but also depends on the sum of the processing times of jobs already processed. They provided the optimal schedules for some single machine problems. Janiak and Rudek (2010) presented a new approach called multi-abilities learning that generalizes the existing ones and models. On this basis, they focused on the makespan problem and provided the optimal polynomial time algorithms for some special cases. Lee et al. (2010) investigated a single-machine problem with the learning effect and release times where the objective is to minimize the

* Corresponding author. Tel.: +886 3 928 395863; fax: +886 3 572 9101.
E-mail address: yhchung.iem96g@nctu.edu.tw (Y.-H. Chung).

0925-5273/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.ijpe.2012.08.014

makespan. Chang et al. (2009) studied a single machine scheduling problem, in which the learning/aging effect is considered. The objective is to determine the common due date and the sequence of jobs that minimizes a cost function.

Often numbers of operations have to be done on every job in many manufacturing and assembly facilities (Pinedo, 2002; Tseng and Lin, 2010; Wang et al., 2010; Zhao and Tang, 2012; Shabtay et al., 2012; Sun et al., 2012). However, it is seldom discussed in the general multiple-machine setting, especially without the assumptions of identical processing time on all the machines or dominant machines. Wang and Xia (2005) studied some permutation flowshop problems when the learning effect is present. They provided the worst-case bound of the shortest processing time rule for the makespan and the total completion time problems. They also showed that the makespan and the total completion time problems remain polynomially solvable for two special cases. Xu et al. (2008) provided heuristic algorithms for some permutation flowshop problems. They also analyzed the worst case bounds for the proposed algorithms. Wu and Lee (2009) considered a permutation flowshop scheduling problem to minimize the total completion time. They also analyzed the performance of the existing heuristic algorithms when the learning effect is present.

With the current emphasis of customer service and meeting the promised delivery dates, in this paper we consider a permutation flowshop scheduling problem to minimize the total tardiness with the learning effect. Although the classical problem without the consideration of learning effects has attracted the attention of numerous researchers due to its simple definition, most of the research focused on developing the heuristic or meta-heuristic algorithms due to the complexity of the problem. Recently, Vallada et al. (2008) provided a comprehensive review of the heuristic algorithms. To the best of our knowledge, Kim (1995) and Chung et al. (2006) were the only authors who derived the optimal schedules. In this paper, we provide a branch-and-bound and two heuristic algorithms when the learning effect is present. The rest of the paper is organized as follows. In the next section we describe the formulation of our problem. In Section 3, we construct a branch-and-bound algorithm using an elimination rule and a lower bound to speed up the search for the optimal solution. In Section 4, two heuristic algorithms are proposed to solve this problem. In Section 5, a computational experiment is conducted to evaluate the efficiency of the branch-and-bound algorithm and the performance of the heuristic algorithms. A conclusion is given in the last section.

2. Problem description

There are n jobs and m machines. For each job j, there are associated with m operations O_{1j}, O_{2j}, …, O_{mj} where operation O_{ij} must be processed on machine i, $i=1,2,\ldots,m$. Processing of operation $O_{i+1,\ j}$ can start only after operation O_{ij} is completed. Moreover, we focus on the permutation flowshop case which implies the job sequence is the same in all the machines. The normal processing time of operation O_{ij} is denoted by p_{ij} and the due date of job j is d_j. The actual processing time p_{ijr} of operation O_{ij} is a function of its position in a schedule. That is,

$$p_{ijr}=p_{ij}r^a,\quad i=1,2,\ldots,m;\quad r=1,2,\ldots,n,$$

if it is scheduled in the rth position and $a<0$ is the learning effect.

For a given schedule S, let $C_{ij}(S)$ denote the completion time of job j on machine i, $T_j(S)=\max 0, C_{mj}(S)-d_j$ denote the tardiness of job j, and $C_{i[j]}(S)$ denote the completion time of the job scheduled in the jth position on machine i. The objective of this paper is to find a schedule that minimizes the total tardiness, a widely used performance measure in scheduling literature. That is, we want to find a schedule S^* such that $\sum T_j(S^*)\le \sum T_j(S)$ for any schedule S.

3. A branch-and-bound algorithm

The problem under study is NP-hard since it already is even without the learning effect (Pinedo, 2002). Thus, the branch-and-bound algorithm might be a good way to obtain the optimal solution. In this section, we first provide a dominance property, followed by the lower bound to speed up the search process, and finally the branch-and-bound algorithm.

3.1. Dominance property

Chung et al. (2006) gave a dominance property for the classical problem. In this subsection, we modified the property to take the learning effect into consideration. Before presenting the property, we first state a lemma from Chung et al. (2006).

Lemma 1. $\max 0, a-b\ge \max 0, a-c-\max 0, b-c$ for arbitrary real numbers a, b, and c.

Property 1. Suppose that $S_1=(\sigma_1,\pi)$ and $S_2=(\sigma_2,\pi)$ are two sequences where σ_1 and σ_2 are partial sequences which contains the same set of s jobs. If

$$\sum_{j=1}^{s} T_{[j]}(\sigma_2)-\sum_{j=1}^{s} T_{[j]}(\sigma_1)\ge (n-s)\max 0, \max_{1\le i\le m} C_{i[s]}(\sigma_1)-C_{i[s]}(\sigma_2),$$

then S_1 dominates S_2.

Proof. By definition, the completion time of the nth job of S_1 on machine m is

$$C_{m[n]}(S_1)=\max_{1\le i\le m} C_{i[n-1]}(S_1)+\sum_{l=i}^{m} p_{l[n]}n^a=C_{i_1[n-1]}(S_1)+\sum_{l=i_1}^{m} p_{l[n]}n^a$$

for some i_1 where $1\le i_1\le m$. Similarly, the completion time of the nth job of S_2 on machine m is

$$C_{m[n]}(S_2)=\max_{1\le i\le m} C_{i[n-1]}(S_2)+\sum_{l=i}^{m} p_{l[n]}n^a=C_{i_2[n-1]}(S_2)+\sum_{l=i_2}^{m} p_{l[n]}n^a$$

for some i_2 where $1\le i_2\le m$. Thus, we have

$$\begin{aligned}C_{m[n]}(S_1)-C_{m[n]}(S_2)&=[C_{i_1[n-1]}(S_1)+\sum_{l=i_1}^{m} p_{l[n]}n^a]-[C_{i_2[n-1]}(S_2)+\sum_{l=i_2}^{m} p_{l[n]}n^a]\\ &\le [C_{i_1[n-1]}(S_1)+\sum_{l=i_1}^{m} p_{l[n]}n^a]-[C_{i_1[n-1]}(S_2)+\sum_{l=i_1}^{m} p_{l[n]}n^a]\\ &\le C_{i_1[n-1]}(S_1)-C_{i_1[n-1]}(S_2)\le \max_{1\le i\le m} C_{i[n-1]}(S_1)-C_{i[n-1]}(S_2)\end{aligned}$$

By an induction argument, we have

$$C_{m[j]}(S_1)-C_{m[j]}(S_2)\le \max_{i\le i\le m} C_{i[s]}(S_1)-C_{i[s]}(S_2)\ \text{for}\ j=s+1,\ldots,n \qquad (1)$$

From Lemma 1 and Eq. (1), the difference between the total tardiness of S_1 and S_2 is

$$\begin{aligned}\sum_{j=1}^{n} T_{[j]}(S_2)-\sum_{j=1}^{n} T_{[j]}(S_1)&=\sum_{j=1}^{s} T_{[j]}(S_2)-\sum_{j=1}^{s} T_{[j]}(S_1)-[\sum_{j=s+1}^{n} T_{[j]}(S_1)-\sum_{j=s+1}^{n} T_{[j]}(S_2)]\\ &=\sum_{j=1}^{s} T_{[j]}(S_2)-\sum_{j=1}^{s} T_{[j]}(S_1)-\sum_{j=s+1}^{n}(\max 0, C_{m[j]}(S_1)-d_{[j]}\\ &\quad -\max 0, C_{m[j]}(S_2)-d_{[j]})\ge \sum_{j=1}^{s} T_{[j]}(S_2)-\sum_{j=1}^{s} T_{[j]}(S_1)\\ &\quad -(n-s)\max 0, \max_{1\le i\le m} C_{i[s]}(S_1)-C_{i[s]}(S_2)\end{aligned}$$

It implies that S_1 dominates S_2 and this completes the proof.

Property 1. can be simplified to the case of two adjacent jobs which is stated without proof.

Corollary 1. Let $S_1=(\pi,j_1,j_2,\pi')$ and $S_2=(\pi,j_2,j_1,\pi')$ be two schedules where partial sequence π contains s jobs. If
$T_{j_1}(S_2)+T_{j_2}(S_2)-T_{j_1}(S_1)-T_{j_2}(S_1)\geq(n-s-2)\max 0,\max_{1\leq i\leq m}[C_{kj_2}(S_1)-C_{kj_1}(S_2)]$,
then S_1 dominates S_2.

3.2. A lower bound

In this subsection, a lower bound is established to facilitate the search process of the branch-and-bound algorithm. Let $\theta=(\pi,\pi^c)$ denote a sequence in which π contains s scheduled jobs and π^c contains $n-s$ unscheduled jobs. Without loss of generality, we assume that the job processing times of the $n-s$ unscheduled jobs on machine k are $p_{k(s+1)}\leq p_{k(s+2)}\leq\cdots\leq p_{k(n)}$ when they are arranged in non-decreasing order. We also assume that the due dates of the $n-s$ unscheduled jobs are $d_{(s+1)}\leq d_{(s+2)}\leq\cdots\leq d_{(n)}$ when they are arranged in non-decreasing order. By definition, the completion time of the $(s+1)$th job on machine k is

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

By an induction argument, the completion time of the $(s+1)$th job on machine m is

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

Therefore, we have

$$C_{m[s+1]}(\theta)\geq\max_{1\leq k\leq m}C_{k[s]}(\theta)+(s+1)^a\sum_{l=k}^{m}p_{l[s+1]} \quad (2)$$

Similarly, the completion time of the $(s+2)$th job on machine k is

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

By an induction argument, the completion time of the $(s+2)$th job on machine m is

$$C_{m[s+2]}(\theta)\geq C_{k[s]}(\theta)+p_{k[s+1]}(s+1)^a+(s+2)^a\sum_{l=k}^{m}p_{l[s+2]}.$$

Thus, we have

$$C_{m[s+2]}(\theta)\geq\max_{1\leq k\leq m}C_{k[s]}(\theta)+p_{k[s+1]}(s+1)^a+(s+2)^a\sum_{l=k}^{m}p_{l[s+2]} \quad (3)$$

Using the same argument as to derive Eqs. (2) and (3), it is obtained that the completion time of the $(s+j)$th job on machine m is

$$C_{m[s+j]}(\theta)\geq\max_{1\leq k\leq m}C_{k[s]}(\theta)+\sum_{v=1}^{j-1}p_{k[s+v]}(s+v)^a+(s+j)^a\sum_{l=k}^{m}p_{l[s+j]} \quad (4)$$

for $1\leq j\leq n-s$. From Eq. (4), it implies that the total tardiness of sequence θ is

$$\begin{aligned}\sum_{j=1}^{n}T_j(\theta)&=\sum_{j=1}^{s}T_j(\theta)+\sum_{j=1}^{n-s}\max 0,C_{m[s+j]}(\theta)-d_{[s+j]}\geq\sum_{j=1}^{s}T_j(\theta)\\&\quad+\sum_{j=1}^{n-s}\max 0,\max_{1\leq k\leq m}C_{k[s]}(\theta)+\sum_{v=1}^{j-1}p_{k[s+v]}(s+v)^a\\&\quad+(s+j)^a\sum_{l=k}^{m}p_{l[s+j]}-d_{[s+j]}\geq\sum_{j=1}^{s}T_j(\theta)+\sum_{j=1}^{n-s}\max 0,\max_{1\leq k\leq m}C_{k[s]}(\theta)\\&\quad+\sum_{v=1}^{j-1}p_{k(s+v)}(s+v)^a+(s+j)^a\min_{i\in\pi^c}\sum_{l=k}^{m}p_{li}-d_{(s+j)}\end{aligned}$$

Thus, the lower bound on the total tardiness of sequence θ based on s scheduled jobs is

$$LB(\theta)=\sum_{j=1}^{s}T_j(\theta)+\sum_{j=1}^{n-s}\max 0,\max_{1\leq k\leq m}C_{k[s]}(\theta)+\sum_{v=1}^{j-1}p_{k(s+v)}(s+v)^a+(s+j)^a\min_{i\in\pi^c}\sum_{l=k}^{m}p_{li}-d_{(s+j)}.$$

3.3. Description of the branch-and-bound algorithm

A depth-first search is adopted in the branching procedure. This method has the advantage that it only requires less storage space. In this paper, the algorithm assigns jobs in a forward manner starting from the first position. In the searching tree, we choose a branch and systematically work down the tree until we either eliminate it by virtue of the dominance property, the lower bound or reach its final node, in which case this sequence either replace the initial solution or is eliminated. The outline of the branch-and-bound algorithm is described as follows.

Step 1. {Initialization} Implement the proposed heuristic algorithm to obtain a sequence as the initial incumbent solution.
Step 2. {Reduction} Apply Corollary 1 to eliminate the dominated partial sequence.
Step 3. {Branching} For the non-dominated nodes, compute the lower bound on the total tardiness of the unfathomed partial sequences or the total tardiness of the completed sequences. If the lower bound on the total tardiness for the partial sequence is greater than the initial solution, eliminate that node and all the nodes beyond it in the branch. If the value of the completed sequence is less than the initial solution, replace it as the new solution. Otherwise, eliminate it.

4. The heuristic algorithms

The computation effort can be reduced by using a heuristic solution as an upper bound prior to the application of the branch-and-bound algorithm. Furthermore, the search for the optimal solution for a problem with a large number of jobs is time consuming, but an effective heuristic can provide a time-saving approximate solution with a small margin of error. Recently, Vallada et al. (2008) provided a review and comprehensive evaluation of

Table 1
Data of the demonstrative examples with $a=-0.152$.

Problem 1						
p_{ij}		j				
		1	2	3	4	5
1		92	71	32	81	74
i	2	29	42	74	93	67
3		5	4	50	70	99
Due date		245	491	373	315	386

Problem 2						
p_{ij}		j				
		1	2	3	4	5
1		22	1	1	62	99
i	2	90	64	14	3	58
3		33	28	52	77	42
Due date		265	279	284	121	196

Problem 3						
p_{ij}		j				
		1	2	3	4	5
1		88	1	64	9	34
i	2	84	36	1	54	41
3		77	75	87	95	66
Due date		437	167	238	257	376

Table 2
The error percentages for the problems with different weights.

	Optimal	$w=0$		$w=0.5$		$w=1$	
	Schedule (total tardiness)	Schedule (total tardiness)	Error (%)	Schedule (total tardiness)	Error (%)	Schedule (total tardiness)	Error (%)
Problem 1	3, 4, 1, 5, 2 (11.93)	4, 1, 3, 5, 2 (14.68)	23.05	4, 1, 3, 5, 2 (14.68)	23.05	3, 4, 1, 5, 2 (11.93)	0
Problem 2	3, 4, 2, 5, 1 (54.26)	3, 2, 4, 1, 5 (73.96)	36.31	3, 4, 2, 5, 1 (54.26)	0	3, 2, 4, 1, 5 (73.96)	36.31
Problem 3	2, 3, 4, 5, 1 (13.69)	2, 3, 4, 5, 1 (13.69)	0	2, 4, 3, 1, 5 (33.12)	141.93	2, 4, 3, 1, 5 (33.12)	141.93

40 different heuristics and metaheuristics for the classical m-machine permutation flowshop problem to minimize the total tardiness. They found that *ENS*2 proposed by Kim et al. (1996) is one of the best heuristic among the existing algorithms. The main idea of algorithm *ENS*2 is modified from Nawaz et al. (1983) by applying the EDD rule instead, and further improving the solution by the pairwise interchange movement. Thus, the algorithm *ENS*2 is used as the base for subsequently analysis.

4.1. A weight combination search algorithm

In our preliminary tests, it was observed that a job should be scheduled in an earlier position if it has a small value of the sum of the job processing times $\sum p_{ij}$ or a small value of due date d_j. This motivated the usage of a combination of these two factors. That is, we would choose a proper weight w and arrange the jobs in a non-decreasing order on the values of $w\sum_{i=1}^{m} p_{ij}+(1-w)d_j$. However, it was very difficult to find the weight that could yield good solutions for all the problems, as illustrated by the examples in Table 1. In problem 1, the optimal solution was yielded when $w=1$, but the error percentages were both 23.05% while$w=0$ and $w=0.5$. In problem 2, the optimal solution was yielded when $w=0.5$ and the error percentages were both 36.31% for the other two cases. In problem 3, the optimal solution was yielded when $w=0$, but the error percentages were both 141.93% for the other two cases. The results were recorded in Table 2. This motivated the usage of a range of the weights, and the near-optimal solution was chosen as the best one among the solutions yielded from different weights. In our preliminary tests with 12 jobs, it was found that the quality of the solutions was quite stable after a partition of (0, 1) into 20 points. The proposed heuristic algorithm is denoted as WS_{NEH+PI}, and the procedures are given as follows.

WS_{NEH+PI} *algorithm* :

Step 1. Set $w=0$, $S^*=(-,\cdots,-)$ with a total tardiness of ∞.
Step 2. Arrange jobs in the non-decreasing order of $w\sum_{i=1}^{m} p_{ij}+(1-w)d_j$. Let US denote the resulting sequence.
Step 3. Set $k=1$, select the first job in US to create a partial sequence *PS*.
Step 4. Update $k=k+1$. Select the kth job from US and insert it in k possible positions in the current partial sequence *PS*. Among k sequences, select the one with the minimum total tardiness as the current partial sequence *PS*.
Step 5. If $k=n$, then replace S^* by *PS* if the total tardiness of *PS* is smaller than that of S^*, else go to Step 4.
Step 6. If $w<1$, set $w=w+0.05$ and go to Step 2.
Step 7. Set $k=1$.
Step 8. If $k<n$, set $l=k+1$ and go to Step 9. Otherwise, stop and output the sequence S^*.
Step 9. Create a new sequence S by exchanging the jobs in positions k and l in S^*. Replace S^* by S if the total tardiness of S is smaller than that of S^*.
Step 10. If $l<n$, then set $l=l+1$ and go to Step 9. Otherwise, set $k=k+1$ and go to Step 8.

4.2. The simulated annealing algorithm

The simulated annealing (*SA*) algorithm, proposed by Kirkpatrick et al. (1983), was among the most popular metaheuristic algorithms. The advantage of *SA* algorithm was that it could avoid getting trapped in a local optimum. In this section, the *SA* algorithm was utilized to derive a near-optimal solution. A brief description of the *SA* procedure was as follows. Given an initial sequence, a new sequence is created by a random neighborhood generation. The new sequence is accepted if its objective function has a smaller value than that of the original sequence; otherwise, it is accepted with some probability that decreases as the process evolves. The temperature is initially set to a high level so that a neighborhood exchange happens frequently in early iterations. It is gradually lowered using a predetermined cooling schedule so that it becomes more difficult to exchange in later iterations unless a better solution is obtained.

The most important implementations of the *SA* algorithm included:

(1) **Initial sequence:** As pointed out by Vallada et al. (2008), *ENS*2 proposed by Kim et al. (1996) was one of the best heuristic among the existing algorithms. Thus, it was used as the initial sequence.
(2) **Neighborhood generation:** Neighborhood generation plays an important role in the efficiency of the *SA* method. Three neighborhood generation methods were used in the preliminary trials. They are the pairwise interchange (*PI*), the extraction and forward-shifted reinsertion (EFSR), and the extraction and backward-shifted reinsertion (EBSR) movements. It was observed that the *PI* movement yielded a better solution in the preliminary trials. Thus, it was used in subsequently analysis.
(3) **Acceptance probability:** In *SA*, solutions are accepted according to the magnitude of increase in the objective function and the temperature. The probability of acceptance is generated from an exponential distribution,

$$P(accept)=\exp(-\alpha\times\Delta TC),$$

where α is the control parameter and ΔTC is the change in the objective function. In addition, the method of changing α at the kth iteration is obtained from Ben-Arieh and Maimon (1992) and is given by

$$\alpha=\frac{k}{\beta}$$

where β is an experimental factor. After some pretests, we chose $\beta=85{,}000$. If the total tardiness increases as a result of a random pairwise interchange, the new sequence is accepted when $P(accept)>r$, where r is a uniform random number between 0 and 1.
(4) **Stopping condition:** Our preliminary tests showed that the schedule is quite stable after $800n$ iterations, where n is the

number of jobs. Thus, $800n$ was used as the number of iterations.

5. Computational experiments

In order to evaluate the performance of the branch-and-bound and the heuristic algorithms, a computational experiment was conducted. All the proposed algorithms were coded in Fortran 90 and run on a personal computer with 2.66 GHz Intel Core 2 Quad CPU Q9400 and 3.25 GB RAM under Windows XP. The normal processing times on the machines were generated from a uniform distribution over the integers 1 to 99 as it was common in the literature. The due dates were generated from another uniform distribution over the integers between $T(1-\tau-R/2)$ and $T(1-\tau+R/2)$, where R is the due date range, τ is the tardiness factor, and T is $\max_{1\le i\le m}\sum_{j=1}^{n} p_{ij} + \min_{1\le j\le n}(\sum_{l=1}^{i-1} p_{lj}) + \min_{1\le j\le n}(\sum_{l=i+1}^{m} p_{lj})$,which is a lower bound of the makespan (1995).

The computational experiment consisted of three parts. In the first part, the job size was fixed at 10 and the numbers of machines were $m=3$ and 5. The due date factors (τ,R) took the values of (0.4, 0.8), (0.4, 1.0), (0.5, 0.8) and (0.5, 1.0) and the values of the learning effects were taken to be 90% and 80%, which corresponded to $a=-0.152$ and -0.322 according to Biskup's (1999) model. To test the efficiency of the dominance property and the lower bound separately, the branch-and-bound algorithm with only the dominance property was denoted as BB_p, the branch-and-bound algorithm with only the lower bound was denoted as BB_L, and the branch-and-bound algorithm with both the property and the lower bound was denoted as BB_{P+L}. The results were compared with the enumeration method. The mean and maximum number of nodes and the mean and maximum CPU time (in seconds) were reported for the branch-and-bound algorithms, while only the mean and maximum CPU time (in seconds) were given for the enumeration method. As a consequence, there were 16 experimental conditions examined in the first part of the experiment, and 100 replications were randomly generated for each condition. The results were presented in Table 3. It was seen that the dominance property and the lower bound are efficient in the searching process for the optimal solution. Moreover, it was noted that the lower bound is more efficient than the dominance property in terms of the execution time and the number of nodes. It was also noticed that the performance of the property is not influenced by the number of machines, while the lower bound is more powerful as the number of machines increases.

In the second part of the experiment, the impacts of the tardiness and the range factors were studied. The number of jobs was fixed at 12, the number of machines was fixed at 5, and the learning effect was 90%. As in Kim et al. (1996), the values of τ and R ranged from 0.1 to 0.5 and from 0.8 to 1.8. The combinations of τ and R were selected in a way that the due dates generated were nonnegative. As a result, 20 combinations of τ and R were examined. 100 replications were generated for each situation, and the results were given in Table 4. When the range factor was fixed, it was seen that the problems are harder to solve as the tardiness factor increases. On the other hand, there was no significant trend on the range factor when the tardiness factor was fixed. Thus, we would choose the last 5 values of τ and R in subsequently experiments.

In the last part of the computational experiments, three different job sizes ($n=14$, 16, and 18) and two numbers of machines ($m=3$ and 5) were tested. The values of the learning effects were chosen to be 90% and 80%. The combination of (τ,R) took the values of (0.4, 0.8), (0.4, 1.0), (0.4, 1.2), (0.5, 0.8), and (0.5, 1.0). As a consequence, 60 experimental situations were examined. A set of 100 instances were randomly generated for each situation, and the results were presented in Tables 5–7. The same sets of instances were used to test the performance of the branch-and-bound and the heuristic algorithms. For the branch-and-bound algorithm, the mean and the maximum execution times (in seconds) as well as the mean and the maximum number of nodes were reported. It was noted in Table 7 that the branch-and-bound algorithm was terminated if the number of nodes explored was over 10^8, which was approximately 6 h in terms of the execution time. The instances with more than 10^8 nodes were denoted as unsolvable instances (USI), which were also reported in Table 7. For the heuristic algorithms, the mean and the maximum error percentages were noted. The error percentage of the solution produced by the heuristic algorithm is calculated as

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

where V is the total tardiness of the solution generated by the heuristic method and V^* is the total tardiness of the optimal schedule. It was noticed that there might be some instances where

Table 3
Performance of the branch-and-bound algorithms and the enumeration method ($n=10$).

m	τ	R	A (%)	Number of nodes						CPU time							
				BB_P		BB_L		BB_{P+L}		BB_P		BB_L		BB_{P+L}		Enumeration	
				Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max
3	0.4	0.8	90	86,445.2	306,377	1876.1	21,474	1390.5	17,501	1.23	4.16	0.07	0.63	0.05	0.58	13.57	13.81
			80	58,910.3	467,754	348.8	4,985	286.5	4,195	0.80	5.69	0.02	0.19	0.01	0.19	13.49	13.69
		1.0	90	99,352.7	336,606	1199.5	17,138	860.4	13,904	1.38	4.48	0.04	0.52	0.04	0.47	13.58	13.83
			80	84,013.7	478,202	327.4	3,179	255.6	2,783	1.13	5.77	0.01	0.13	0.01	0.13	13.51	13.72
	0.5	0.8	90	101,868.8	499,679	2463.1	17,206	1558.6	11,418	1.43	6.81	0.09	0.50	0.06	0.45	13.54	13.70
			80	73,966.9	294,852	507.3	7,392	377.9	5,192	1.03	3.70	0.02	0.23	0.02	0.19	13.50	13.72
		1.0	90	98,875.7	271,798	1635.0	11,577	1036.0	5,074	1.38	3.89	0.06	0.36	0.04	0.20	13.55	13.73
			80	89,270.3	280,130	413.3	5,962	286.6	3,026	1.22	3.56	0.02	0.17	0.01	0.09	13.50	13.69
5	0.4	0.8	90	148,516.4	516,225	1289.7	10,438	1052.3	7,564	3.27	10.89	0.08	0.48	0.07	0.44	21.81	22.16
			80	96,536.0	420,392	289.9	3,032	264.5	2,919	2.10	8.73	0.02	0.19	0.02	0.20	21.68	21.91
		1.0	90	160,427.6	501,092	897.7	8,093	732.2	5,953	3.48	10.08	0.05	0.41	0.05	0.34	21.83	22.11
			80	130,483.0	524,809	259.0	2,339	236.0	2,242	2.77	10.64	0.02	0.14	0.02	0.16	21.69	21.94
	0.5	0.8	90	222,501.4	661,605	1965.9	14,354	1552.8	9,528	4.74	13.16	0.11	0.56	0.10	0.42	21.78	22.02
			80	122,595.4	605,737	390.4	3,656	338.0	3,181	2.66	11.77	0.03	0.22	0.03	0.22	21.66	21.98
		1.0	90	220,272.3	779,452	1503.3	12,731	1194.4	8,584	4.67	15.02	0.08	0.52	0.08	0.45	21.80	22.08
			80	143,913.9	383,365	335.5	2,711	293.1	2,372	3.07	7.98	0.02	0.16	0.02	0.16	21.67	21.97

the tardiness value of the branch-and-bound algorithm is zero but that of the heuristic algorithm is not. In that case, the error percentage was not computed, and denoted as E_∞ in the tables. Thus, the error percentages in Tables 5–7 could only be regarded as the lower bounds of the errors. The computational times of the heuristic algorithms were not recorded since they were finished within a second. It was observed from the tables that the number of nodes and the execution time grow exponentially as the number of jobs increases. There were 5 unsolvable instances out of a total of 2000 when $n=18$. The most time-consuming solvable case took about 4.9 h. It was observed that the impact of the number of machines on the performance of the branch-and-bound algorithm is not significant when the values of the other parameters were fixed. It was seen that the problems are easier to solve as the learning effect is stronger. As stated earlier, the problems are harder to solve as the tardiness factor increases. On the other hand, there was no significant trend on the range factor when the tardiness factor was fixed. As the performance of the heuristic algorithms, it was seen from the error percentage of *ENS2* that the problems are harder when the learning effect is considered. It was observed that both the weight combination and the *SA* approaches significantly improved the quality of the near-optimal solutions. However, there was no absolutely dominance relation between the performance of the WS_{NEH+PI} and the *SA* approaches. It was worth to mentioned that out of the 5995 solvable instances, *ENS2* has 33 instances that the tardiness value of the branch-and-bound algorithm is zero but that of the heuristic algorithm is not, WS_{NEH+PI} has 7 instances, and *SA* has only 2 instances. Although the error percentages of the WS_{NEH+PI} and the *SA* algorithms were still high for some instances, it was due to the fact that the optimal tardiness values are relatively small. For instance, the tardiness values of the WS_{NEH+PI} and the *SA* algorithms were 5.56 and 9.56, while the optimal tardiness value was 2.055. This yielded an error percentage of 170.56% and 365.21%, respectively when $n=16$, $m=3$, $(\tau,R)=(0.4, 0.8)$, and the learning effect was 90%. The overall performances of the WS_{NEH+PI} and the *SA* algorithms were quite good and they were recommended when the learning effect was present.

Table 4
Performance of the branch-and-bound algorithm with respect to τ and R ($n=12, m=5$ and LE=90%).

τ	R	Branch-and-bound algorithm			
		Number of nodes		CPU time	
		Mean	Max	Mean	Max
0.1	0.8	907.7	43,155	0.097	3.938
0.1	1.0	1,633.4	110,311	0.168	10.031
0.1	1.2	300.7	7,317	0.042	0.922
0.1	1.4	401.1	13,319	0.053	1.484
0.1	1.6	954.0	37,936	0.110	3.719
0.1	1.8	1,198.4	56,129	0.133	4.953
0.2	0.8	3,323.0	117,571	0.343	11.844
0.2	1.0	2,572.4	34,737	0.262	3.094
0.2	1.2	899.2	17,019	0.110	1.766
0.2	1.4	1,233.5	40,551	0.133	3.203
0.2	1.6	915.8	25,563	0.114	2.797
0.3	0.8	4,324.9	67,270	0.435	6.031
0.3	1.0	2,727.8	29,106	0.291	2.938
0.3	1.2	2,652.3	38,465	0.284	3.250
0.3	1.4	2,535.7	54,157	0.266	4.453
0.4	0.8	8,463.7	76,582	0.779	5.969
0.4	1.0	6,416.3	68,311	0.617	6.078
0.4	1.2	2,044.0	20,851	0.226	1.750
0.5	0.8	12,890.6	95,669	1.153	7.938
0.5	1.0	9,212.9	83,816	0.840	6.109

6. Conclusion

In this paper, we studied an m-machine permutation flowshop problem to minimize the total tardiness with the learning effect. The computational experiments showed that the dominance rule and the lower bound facilitate the search for the optimal solution. The results also showed that the proposed branch-and-bound algorithm could solve most of the problems with up to 18 jobs in a reasonable amount of time. In addition, computational experiments showed that the performance of the proposed heuristics were good for instances of up to 18 jobs.

Table 5
Performance of the branch-and-bound and the heuristic algorithms ($n=14$).

m	a	τ	R	Branch-and-bound algorithm				Lower bound on the error percentage of the heuristics						E_∞		
				Number of nodes		CPU time		*ENS2*		WS_{NEH+PI}		*SA*		*ENS2*	WS_{NEH+PI}	*SA*
				Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max			
3	90%	0.4	0.8	31,665.4	745,932	2.8	59.8	4.43	66.66	0.81	30.91	0.45	32.42	2	0	0
		0.4	1.0	16,479.1	320,261	1.5	23.4	5.68	178.62	0.39	14.77	0.33	25.61	1	0	0
		0.4	1.2	29,930.9	918,455	2.4	71.6	2.65	35.44	0.46	10.70	0.25	9.58	0	0	0
		0.5	0.8	164,189.2	3936,793	12.0	274.9	10.37	364.86	4.68	364.86	2.39	198.56	0	0	0
		0.5	1.0	27,701.1	494,319	2.4	40.3	4.09	92.53	0.31	5.27	0.13	4.00	0	0	0
	80%	0.4	0.8	14,133.8	1290,356	1.0	83.7	5.48	148.65	0.47	44.98	0.04	4.18	3	1	0
		0.4	1.0	2,243.5	38,408	0.2	4.4	3.92	75.40	0.25	7.64	0.05	3.21	3	0	0
		0.4	1.2	4,122.0	244,220	0.4	16.4	2.45	130.28	0.13	4.13	0.04	2.88	1	0	0
		0.5	0.8	5,607.0	105,935	0.5	8.4	26.79	2,336.72	0.54	12.58	0.07	5.53	0	0	0
		0.5	1.0	13,962.1	247,802	1.1	19.1	1.89	18.08	0.16	3.97	0.05	3.37	0	0	0
5	90%	0.4	0.8	67,788.0	2218,760	7.8	152.8	10.77	166.27	2.06	27.16	0.73	11.27	0	0	0
		0.4	1.0	32,581.3	433,044	4.4	48.7	4.81	38.03	0.45	4.81	0.42	5.35	0	0	0
		0.4	1.2	13,864.8	322,278	2.1	41.5	2.23	18.44	0.54	9.28	0.22	3.22	0	0	0
		0.5	0.8	78,784.8	850,160	10.1	93.1	6.21	47.95	1.70	30.28	0.60	8.80	0	0	0
		0.5	1.0	40,729.5	552,293	5.5	66.5	3.12	20.57	0.92	9.04	0.43	5.75	0	0	0
	80%	0.4	0.8	3,740.1	50,277	0.7	8.5	135.48	12,580.30	0.85	53.30	0.86	34.55	1	0	0
		0.4	1.0	2,582.3	27,599	0.5	4.6	3.88	126.19	0.30	9.09	0.10	4.85	0	0	0
		0.4	1.2	2,306.7	38,603	0.4	5.2	44.53	4,037.97	22.38	2138.06	14.63	1453.15	0	0	0
		0.5	0.8	7,614.9	89,364	1.1	11.5	2.48	27.15	0.48	8.31	0.29	8.02	0	0	0
		0.5	1.0	4,257.0	62,313	0.7	10.0	2.72	63.46	0.39	8.43	0.30	6.90	0	0	0

Table 6
Performance of the branch-and-bound and the heuristic algorithms ($n=16$).

m	a	τ	R	Branch-and-bound algorithm				Lower bound on the error percentage of the heuristics						E_∞		
				Number of nodes		CPU time		*ENS*2		WS_{NEH+PI}		*SA*		*ENS*2	WS_{NEH+PI}	*SA*
				Mean	Nax	Mean	Max	Mean	Max	Mean	Max	Mean	Max			
3	90%	0.4	0.8	317,003.3	13,430,827	33.4	1,102.5	11.92	425.99	2.20	170.56	4.02	365.21	1	0	0
		0.4	1.0	902,345.2	62,734,480	84.0	5,486.7	4.17	48.75	0.43	8.62	0.57	16.70	1	0	0
		0.4	1.2	35,242.6	772,007	4.6	85.6	1.98	46.37	0.47	7.28	0.21	5.56	0	0	0
		0.5	0.8	488,873.8	13,790,144	51.9	1,155.7	7.14	85.12	0.96	17.04	1.47	44.90	2	1	0
		0.5	1.0	984,992.0	48,954,248	92.5	4,217.7	3.64	77.19	1.16	76.91	0.27	4.20	0	0	0
	80%	0.4	0.8	8,214.9	219,510	1.3	34.2	6.24	133.90	1.15	106.50	0.19	9.87	1	0	0
		0.4	1.0	19,829.1	1,030,480	2.7	122.9	4.31	121.02	0.09	5.12	0.42	14.88	0	0	0
		0.4	1.2	9,718.4	344,404	1.4	44.2	2.32	99.44	0.96	83.31	0.05	2.15	0	0	0
		0.5	0.8	19,266.9	549,328	2.7	70.0	8.13	289.41	1.14	19.56	0.33	12.86	0	0	0
		0.5	1.0	41,233.2	578,872	5.2	67.8	2.20	33.10	0.75	28.91	0.17	7.62	0	0	0
5	90%	0.4	0.8	340,616.4	17,409,072	60.0	2,766.1	17.25	655.79	7.53	553.15	1.33	17.41	0	0	0
		0.4	1.0	169,525.6	4,801,822	32.5	804.9	5.28	73.63	1.24	10.77	1.35	55.72	0	0	0
		0.4	1.2	151,007.9	6,301,920	27.4	931.5	2.33	15.05	0.28	6.32	0.25	6.32	0	0	0
		0.5	0.8	1,091,356.8	12,536,075	181.8	2,137.6	7.43	37.60	1.98	14.60	1.37	16.67	0	0	0
		0.5	1.0	1,138,042.9	76,265,448	180.6	11,481.7	4.88	43.04	1.04	9.42	0.74	14.66	0	0	0
	80%	0.4	0.8	5,666.1	115,270	1.5	25.8	13.60	223.82	3.42	143.14	0.41	12.79	0	0	0
		0.4	1.0	9,055.9	231,672	2.1	52.6	4.16	65.41	0.26	9.50	0.22	6.73	1	0	0
		0.4	1.2	14,295.0	191,760	3.4	39.5	3.32	180.18	0.36	7.12	0.31	12.56	0	0	0
		0.5	0.8	33,986.5	805,828	7.2	170.6	5.84	115.27	1.20	37.41	0.48	15.47	0	0	0
		0.5	1.0	20,231.7	436,435	4.5	82.5	2.77	28.83	0.30	5.82	0.28	4.35	0	0	0

Table 7
Performance of the branch-and-bound and the heuristic algorithms ($n=18$).

m	a	τ	R	Branch-and-bound algorithm				Lower bound on the error percentage of the heuristics						E_∞			USI
				Number of nodes		CPU time		*ENS*2		WS_{NEH+PI}		*SA*		*ENS*2	WS_{NEH+PI}	*SA*	
				Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max				
3	90%	0.4	0.8	713,301.6	19,509,458	112.2	2,739.4	309.17	26,525.00	12.33	1089.64	21.63	1896.62	2	1	0	0
		0.4	1.0	1,193,123.5	59,651,120	192.2	9,211.3	14.12	602.08	4.07	340.34	0.79	54.74	3	0	0	0
		0.4	1.2	368,085.7*	9,941,654*	68.1*	1,931.5*	2.79	50.51	0.75	39.27	0.15	5.06	0	0	0	1
		0.5	0.8	3,807,059.3*	69,093,792*	499.2*	9,345.5*	17.82	798.11	2.44	109.39	2.14	109.39	2	0	0	2
		0.5	1.0	1,464,720.8*	38,858,224*	218.2*	5,040.9*	2.56	37.01	0.53	7.14	0.51	10.28	0	0	0	1
	80%	0.4	0.8	99,529.5	7,693,594	18.0	1,308.8	1.53	63.88	0.00	0.00	0.02	1.72	1	0	0	0
		0.4	1.0	63,843.9	2,911,862	12.3	532.1	5.26	114.00	0.24	5.34	0.18	7.79	1	1	0	0
		0.4	1.2	110,169.3	8,875,314	18.1	1,338.4	1.55	23.17	0.15	6.09	0.21	7.35	0	0	0	0
		0.5	0.8	198,012.8	9,168,472	33.7	1,389.6	10.11	282.92	0.34	9.75	0.76	53.80	0	0	0	0
		0.5	1.0	364,668.3	29,460,512	61.1	4,730.8	3.25	70.22	0.90	31.28	0.43	17.92	0	0	0	0
5	90%	0.4	0.8	693,555.3*	9,369,027*	182.4*	2,280.3*	11.96	364.93	3.53	123.37	3.58	163.08	2	1	2	1
		0.4	1.0	234,498.1	4,391,182	67.2	1,279.4	10.13	479.05	3.84	276.03	0.78	30.40	0	0	0	0
		0.4	1.2	442,895.6	14,758,786	120.3	3,244.0	2.69	32.28	0.58	10.14	0.55	18.29	0	0	0	0
		0.5	0.8	3,615,769.3	83,501,184	828.2	17,656.1	8.53	62.14	1.89	19.04	2.60	62.14	0	0	0	0
		0.5	1.0	3,187,206.0	50,813,748	764.4	12,792.8	5.34	66.97	1.37	19.85	0.84	19.14	0	0	0	0
	80%	0.4	0.8	25,074.3	753,083	7.6	242.0	10.99	191.43	0.40	20.27	0.89	48.92	3	2	0	0
		0.4	1.0	97,149.0	7,736,601	24.8	1,816.3	12.24	624.45	1.17	52.76	6.80	624.45	1	0	0	0
		0.4	1.2	48,316.4	2,568,162	14.	668.4	2.61	30.63	0.38	11.06	0.36	12.01	0	0	0	0
		0.5	0.8	54,895.8	1,435,597	16.0	332.0	5.44	100.05	0.68	19.06	0.43	13.91	1	0	0	0
		0.5	1.0	99,501.8	1,743,517	29.9	435.4	2.91	31.20	0.66	15.54	0.28	4.68	0	0	0	0

* These are the lower bounds on the mean values or max values because of the unsolvable instances in these sets.

Acknowledgements

The authors are grateful to the editor and the referees, whose constructive comments have led to a substantial improvement in the presentation of the paper. This work was supported by the NSC of Taiwan, ROC, under NSC 98–2221-E-035-033-MY2.

References

Ben-Arieh, D., Maimon, O., 1992. Annealing method for PCB assembly scheduling on two sequential machines. International Journal of Computer Integrated Manufacturing 5, 361–367.

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.

Chang, P.C., Chen, S.H., Mani, V., 2009. A note on due-date assignment and single machine scheduling with a learning-aging effect. International Journal of Production Economics 117, 142–149.

Cheng, T.C.E., Wang, G., 2000. Single machine scheduling with learning effect considerations. Annals of Operations Research 98, 273–290.

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.

Cheng, T.C.E., Wu, C.C., Chen, J.C., Wu, W.H., Cheng, S.R. Two-machine flowshop scheduling with a truncated learning function to minimize the makespan. International Journal of Production Economics, http://dx.doi.org/10.1016/j.ijpe.2012.03.027, In press.

Chung, C.S., Flynn, J., Kirca, O., 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.

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-abilities learning effect. Omega 38, 213–217.
Kim, Y.D., 1995. Minimizing total tardiness in permutation flowshops. European Journal of Operational Research 85, 541–555.
Kim, Y.D., Lim, H.G., Park, M.W., 1996. Search heuristics for a flowshop scheduling problem in a printed circuit board assembly process. European Journal of Operational Research 91, 124–143.
Kirkpatrick, S., Gelatt, C., Vecchi, M., 1983. Optimization by simulated annealing. Science 220, 671–680.
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 effect scheduling problem with release times. Omega 38, 3–11.
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, second ed.. Prentice-Hall, Englewood Cliffs, NJ.
Shabtay, D., Bensoussan, Y., Kaspi, M., 2012. A bicriteria approach to maximize the weighted number of just-in-time jobs and to minimize the total resource consumption cost in a two-machine flow-shop scheduling system. International Journal of Production Economics 136, 67–74.
Sun, L.H., Sun, L.Y., Wang, M.Z., Wang, J.B., 2012. Flow shop makespan minimization scheduling with deteriorating jobs under dominating machines. International Journal of Production Economics 138, 195–200.
Tseng, L.Y., Lin, Y.T., 2010. A genetic local search algorithm for minimizing total flowtime in the permutation flowshop scheduling problem. International Journal of Production Economics 127, 121–128.
Vallada, E., Ruiz, R., Minella, G., 2008. Minimising total tardiness in the m-machine flowshop problem: a review and evaluation of heuristics and metaheuristics. Computers and Operations Research 35, 1350–1373.
Wang, J.B., 2007. Single-machine scheduling problems with the effects of learning and deterioration. Omega 35, 397–402.
Wang, J.B., 2008. Single-machine scheduling with general learning functions. Computers and Mathematics with Applications 56, 1941–1947.
Wang, L., Sun, L.Y., Sun, L.H., Wang, J.B., 2010. On three-machine flow shop scheduling with deteriorating jobs. International Journal of Production Economics 125, 185–189.
Wang, J.B., Xia, Z.Q., 2005. Flow-shop scheduling with a learning effect. Journal of 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.
Xu, Z.Y., Sun, L.Y., Gong, J.T., 2008. Worst-case analysis for flow shop scheduling with a learning effect. International Journal of Production Economics 113, 748–753.
Yin, Y.Q., Xu, D.H., Sun, K.B., Li, H.X., 2009. Some scheduling problems with general position-dependent and time-dependent learning effects. Information Sciences 179, 2416–2425.
Zhang, X.G., Yan, G., 2010. Machine scheduling problems with a general learning effect. Mathematical and Computer Modelling 51, 84–90.
Zhao, C.L., Tang, H.Y., 2012. Two-machine flow shop scheduling with deteriorating jobs and chain precedence constraints. International Journal of Production Economics 136, 131–136.