Int J Adv Manuf Technol (2011) 56:355–367
DOI 10.1007/s00170-011-3172-2

ORIGINAL ARTICLE

Makespan minimization for *m*-machine permutation flowshop scheduling problem with learning considerations

Yu-Hsiang Chung · Lee-Ing Tong

Received: 14 September 2010 / Accepted: 17 January 2011 / Published online: 8 February 2011

Abstract Studies on scheduling with learning considerations have recently become important. Most studies focus on single-machine settings. However, numerous complex industrial problems can be modeled as flowshop scheduling problems. This paper thus focuses on minimizing the makespan in an *m*-machine permutation flowshop with learning considerations. This paper proposes a dominance theorem and a lower bound to accelerate the branch-and-bound algorithm for seeking the optimal solution. This paper also adapts four well-known existing heuristic algorithms to yield the near-optimal solutions. Eventually, the performances of all the algorithms proposed in this paper are reported for small and large job-sized problems. The computational experiments indicate that the branch-and-bound algorithm can solve problems of up to 18 jobs within a reasonable amount of time, and the heuristic algorithms are quite accurate with a mean error percentage of less than 0.1%.

Keywords Scheduling · Learning effects · Flowshop · Makespan

1 Introduction

In traditional scheduling problems, it is assumed that all the job processing times are fixed and known (Pinedo [1]; Smith [2]). However, job processing times frequently decline as workers gather working knowledge and experience. For example, processing similar tasks continuously improves worker skills and helps workers perform their jobs efficiently (Biskup [3]). This phenomenon is known as the "learning effect." The influence of learning on productivity for aircraft industry manufacturing was first observed by Wright [4] and subsequently affirmed in numerous industries such as the manufacturing and service industries (Yelle [5]).

Biskup [3] introduced a learning effect scheduling model in which the actual processing time of a job decreases when the job is scheduled late. He examined the problems associated with minimizing the deviation from a common due date and the sum of flow times in a single-machine environment, and demonstrated that the problems are polynomially solvable. Subsequently, numerous studies have considered this novel and extended region. Cheng et al. [6] developed a model with learning effect in which actual job processing time is based on the total normal job processing time and the position of schedule on a single machine. They then demonstrated that the makespan and total completion time problems are polynomially solvable, and demonstrated that the problems for minimizing weighted completion time and maximum lateness are polynomially solvable with certain agreeable conditions. Biskup [7] presented a detailed review of scheduling problems with learning effect. Particularly, he classified the existing models into two distinct groups: the position-based learning and the sum-of-processing-time-based learning. The position-based learning is influenced by the number of jobs processed. Meanwhile, the sum-of-processing-time-based learning considers the processing time of the jobs processed to date.

In the position-based learning model, Wang et al. [8] investigated a single-machine scheduling problem in which the setup time and learning effect are considered, and the setup times are past-sequence-dependent. They showed that

Y.-H. Chung (✉) · L.-I. Tong
Department of Industrial Engineering and Management,
National Chiao Tung University,
Hsinchu, Taiwan, Republic of China
e-mail: yhchung.iem96g@nctu.edu.tw

the problems to minimize the sum of quadratic job completion time, the total waiting time, the total absolute differences in waiting time, and the sum of earliness penalties subject to no tardy jobs, are polynomially solvable. Wang et al. [9] studied a single-machine problem with learning effect and discounted cost. They showed that the shortest processing time first (SPT) rule is the optimal policy for minimizing the discounted total completion time. They then illustrated an example to demonstrate that the discounted weighted shortest processing time first rule is not the optimal policy for minimizing the discounted total weighted completion time. Furthermore, Janiak and Rudek [10] proposed a new learning effect model in which the rigorous constraints of the position-dependent approach are relaxed by assuming that each job creates a different experience for the processor. They also described the shape of the learning curve using a *k*-stepwise function. Hence, the diversified learning functions can be fitted by a mathematical model. Janiak and Rudek [11] proposed a new experience-based learning model where the job processing times are described by "*S*"-shaped functions and are dependent on the experience of the processor. They demonstrated that the makespan problem on a single processor is NP-hard or strongly NP-hard and then provided a number of polynomially solvable cases. In addition, Toksari and Guner [12] considered a parallel machine earliness/tardiness scheduling problem involving different penalties under the effect of position-based learning and deterioration, and demonstrated that the optimal solution is a V-shaped schedule under certain agreeable conditions. Eren and Güner [13] studied a bicriteria scheduling problem with a learning effect in an *m*-identical parallel machine environment, and the objective function is to minimize the weighted sum of the total completion time and total tardiness. They constructed a mathematical programming model to solve the problem.

As for the sum-of-processing-time-based learning model, Koulamas and Kyparisis [14] pointed out that employees learn more when executing jobs with a longer processing time. They introduced a sum-of-job-processing-time-based learning effect scheduling model and demonstrated that the makespan and the total completion time problems for the single-machine and two-machine flowshops with ordered job processing times are polynomially solvable. Wu et al. [15] studied a total weighted completion time problem on a single machine with learning effect and ready times. A branch-and-bound algorithm was proposed to derive the optimal solution, and the simulated annealing algorithm was implemented to obtain the near-optimal solution. Furthermore, Cheng et al. [16] introduced a learning effect model on a single machine in which the actual job processing time is derived from the sum of the logarithm of the processing times of jobs already processed, and they show that the makespan and total completion time problems are polynomially solvable. Wang et al. [17] demonstrated that, even with the effects of sum-of-processing-time-based learning and deterioration on job processing times, the single-machine makespan problem remains polynomially solvable. Wang et al. [18] considered the weighted sum of completion times and the maximum lateness problem with the effect of learning and deterioration on a single machine where job processing times are defined as functions of their starting times and sequential positions.

In recent literature, the position-based and the sum-of-processing-time-based learning have been discussed simultaneously. Yin et al. [19] examined some single-machine and *m*-machine flowshop problems with learning considerations where the learning effect is not only a function of the total normal processing times of jobs already processed, but also of the scheduled job position. Lee and Wu [20] presented a general learning model that simultaneously combines the position-based learning and sum-of-processing-time-based learning models. They then demonstrated that the single-machine makespan and the total completion time problems are polynomially solvable and provided polynomial-time optimal solutions for minimizing the makespan and total completion time under certain conditions in a flowshop environment.

The concept of learning effect in a flowshop environment has been relatively neglected. However, Wu et al. [21] studied the maximum tardiness problem with the position-based learning effect in a two-machine flowshop environment. They implemented a branch-and-bound algorithm to obtain the optimal solution and a simulated annealing algorithm to obtain the near-optimal solution. In addition, Lee and Wu [22] considered a two-machine flowshop problem with learning effect for minimizing the total completion time. They utilized two lower bounds and several dominance properties to construct a branch-and-bound algorithm to obtain the optimal solution and established a heuristic algorithm to obtain the near-optimal solution. Chen et al. [23] considered a bicriteria two-machine flowshop scheduling problem with the position-based learning effect when the goal is to minimize both the total completion time and the maximum tardiness. They proposed a branch-and-bound algorithm and two heuristic algorithms to obtain the optimal and near-optimal solutions. Furthermore, Wang and Xia [24] studied flowshop problems with learning effect. They gave the worst-case bound of the SPT algorithm for the makespan and the total flow time problems and then illustrated examples to show that the Johnson's rule is not optimal for the makespan problem in a two-machine environment with learning consideration. Eventually, they demonstrated that two special cases remained polynomially solvable for the makespan and total completion time problems. Additionally, Wu and Lee [25] investigated a flowshop problem with learning considerations to minimize total completion time. They implemented a branch-and-bound algorithm and heuristic algorithms to seek the optimal and near-optimal solutions, respectively.

Since obtaining optimal solutions in scheduling problems within a flowshop environment is usually complicated, numerous works have focused on identifying efficient near-optimal solutions. In the literature of multiple machine flowshop without learning effect consideration, Nawaz et al. [26] considered an m-machine flowshop problem for minimizing the makespan and claimed that jobs with larger total normal processing time should be prioritized over jobs with smaller total normal processing times. They demonstrated that their proposed algorithm performs particularly well on large job-sized problems. Furthermore, Liu and Ong [27] and Ruiz and Maroto [28] claimed that the algorithm developed by Nawaz et al. [26] is superior to other existing polynomial algorithms for the m-machine flowshop makespan problem. Rajendran and Ziegler [29] developed an algorithm for solving the weighted total completion time minimization problem in an m-machines flowshop environment. Their algorithm first generates m sequences by assigning different weights to each machine. The sequence with the minimal total weighted completion time is then selected as the seed sequence, and an improvement scheme is employed. Woo and Yim [30] provided an algorithm for minimizing the mean flow time in an m-machine flowshop environment. Their algorithm selects a job among excluded jobs for insertion into the current partial sequence. Whenever a new partial schedule is constructed, their algorithm assesses all the possible sequences by inserting an unscheduled job into one of all slots in the current sequence at a time. The partial sequence with the least mean flow time is selected. Framinan and Leisten [31] considered an m-machine flowshop problem to minimize the mean flow time. They proposed an efficient constructive heuristic algorithm based on the concept of the algorithm of Nawaz et al. [26]. They further performed a general pairwise interchange movement to boost the quality of the partial schedules in all the iterations.

In this paper, we examine the model of Biskup [3] in the m-machine flowshop environment. Garey et al. [32] demonstrated that the flowshop scheduling problem for minimizing the makespan without learning effect is NP-hard. Therefore, the branch-and-bound algorithm is a feasible approach for deriving the optimal solution. In the literature about the flowshop scheduling problem without learning effect, Chung et al. [33] studied an m-machine flowshop problem to minimize the total completion time. They proposed a brand-and-bound algorithm that incorporates an innovative lower bound and a dominance criterion to seek the optimal solution. They then investigated the performances of the brand-and-bound algorithm using six data types. Furthermore, Chung et al. [34] considered a total tardiness scheduling problem in an m-machine flowshop environment. They obtained the optimal solution by utilizing a branch-and-bound algorithm and then compared the algorithm they proposed with the best alternative existing algorithm.

The remainder of this paper is organized as follows. Section 2 details the formulation of the problem. Section 3 then establishes a dominance theorem and a lower bound and modifies four well-known heuristic algorithms to solve the proposed problem. Section 4 conducts a computational experiment to assess the performances of all proposed algorithms. Conclusions are finally drawn in section 5.

2 Notations and problem statement

The notations used throughout this paper are summarized as follows.

n	Number of jobs.
m	Number of machines.
N	Set of jobs, i.e., $N=\{1,2,\ldots,n\}$.
M_i	ith machine, $i=1,2,\ldots,m$.
$p_{i,j}$	Normal processing time of job j on M_i.
$p_{i,j,r}$	Actual processing time of job j on M_i if placed at position r in a schedule.
a	Learning index with $a<0$.
S	Subset of N with s scheduled jobs.
U	Subset of N with $n-s$ unscheduled jobs.
σ	A partial sequence of set S.
[]	The symbol which signifies the order of jobs in a schedule.
$C_{i,[r]}(\sigma)$	Completion time of the job scheduled in the rth position on M_i in sequence σ.
$G_j(u,v)$	Total normal processing time of job j from M_u to M_v, where $u\leq v$, i.e., $G_j(u,v)=\sum_{l=u}^{v}p_{l,j}$.
$B_{i,[r]}$	Earliest starting time at rth position on M_i.
$F_{i,[r]}$	Earliest completion time at rth position on M_i.
LB	The lower bound for the current node.

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

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

where i=1,2,...,m, j,r=1,2,...,n.

This paper attempts to identify a schedule for minimizing the makespan, a widely used performance measure in the scheduling literature. For a given schedule τ with n jobs, the objective of this paper is to derive a schedule τ^* such that $C_{m[n]}(\tau^*) \le C_{m[n]}(\tau)$ for all schedules τ.

3 Algorithms

To facilitate the branch-and-bound algorithm, a dominance theorem and a lower bound are proposed in this section. Furthermore, four well-know heuristic algorithms are modified to yield the near-optimal solution. Finally, the detailed procedure of the proposed branch-and-bound algorithm is represented.

3.1 Dominance theorem of branch-and-bound algorithm

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

Theorem *Let σ_1 and σ_2 denote two partial sequences with s jobs of set S. If $\max_{1\le i\le m}\{C_{i,[s]}(\sigma_1) - C_{i,[s]}(\sigma_2)\} < 0$, then σ_1 dominates σ_2.*

Proof Let π denote a partial sequence with $n{-}s$ jobs of set U, and sequence π is scheduled immediately behind sequence σ_1 and σ_2 into the sequence S_1=(σ_1, π) and S_2=(σ_2, π), respectively. Then, for 1≤u≤m, we have the completion time of the job scheduled in the nth position on M_u in S_1 and is

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

where 1≤v_1≤u.

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

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

where 1≤v_2≤u.

Then, we have $C_{u,[n]}(S_2) \ge C_{v_1,[n-1]}(S_2) + G_{[n]}(v_1,u)\times n^a$ for $v_1 \ne v_2$.

Therefore, we have

$$\begin{aligned} C_{u,[n]}(S_1) - C_{u,[n]}(S_2) \le& \left[C_{v_1,[n-1]}(S_1) + G_{[n]}(v_1,u) \times n^a\right] \\ & - \left[C_{v_1,[n-1]}(S_2) + G_{[n]}(v_1,u) \times n^a\right] \end{aligned}$$

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

An induction argument is conducted. Then, we have

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

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

The proof is completed.

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

Corollary Let J_x and J_y denote two jobs of set S, and σ_{s-2} denote a sequence with s−2 jobs excluding J_x and J_y of set S. If $\max_{1\le i\le m}\{C_{i,[s]}(\sigma_{s-2}, J_x, J_y) - C_{i,[s]}(\sigma_{s-2}, J_y, J_x)\} < 0$, then sequence (σ_{s-2}, J_x, J_y) dominates (σ_{s-2}, J_y, J_x).

3.2 The lower bound of branch-and-bound algorithm

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

In this subsection, we propose a lower bound for eliminating nodes in the branching tree, and the lower bound is evaluated by using the concept developed by Chung et al. [33]. The lower bound for Chung et al. [33] is a machine-based lower bound. The main idea of their lower bound is assuming that the given machine has unit capacity and the machines behind it have infinite capacity. Hence, the procedure in Chung et al. [33] for estimating the marginal lower bound based on the given machine is to compute the earliest starting times for all remaining positions on the machine at first, and to sum up these starting times and all the processing times of the machine and that behind the machine for unscheduled jobs. Finally, the lower bound is determined as the maximal marginal lower bound. Instead of the total completion time, we adapt the procedure in Chung et al. [33] which estimates the earliest starting time with learning effect, when the objective is to minimize the makespan. The proposed lower bound is summarized as follows. Let $p_{i,(j)}$ represent the normal processing times on M_i, which are based on non-descending order of all $p_{i,j}$ from set U for j=1,2,…n−s, i.e., $p_{i,(1)} \leq p_{i,(2)} \leq \cdots \leq p_{i,(n-s)}$, where i=1,2,…,m. $G_{(1)}(u,v)$ denote the smallest total normal processing time between M_u and M_v from set U. Let $E_{i,[s+1]}$ denote the actual starting time of s+1th job on M_i. By definition, we have

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

and

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

where i=2,3,…m.

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

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

$$\leq \max\{B_{1,[s+1]} + p_{1,(1)}\times(s+1)^a, C_{2,[s]}(\sigma)\}.$$

Therefore, $B_{2,[s+1]}$ is evaluated as $\max\{B_{1,[s+1]}+ p_{1,(1)}\times(s+1)^a, C_{2,[s]}(\sigma)\}$. By induction, we have $B_{i,[s+1]} = \max\{\max_{1\leq u\leq i-1}\{B_{u,[s+1]} + G_{(1)}(u,i-1)\times(s+1)^a\},$ $C_{i,[s]}(\sigma)\}$ for i=2, 3,…m.

Since the learning effect is considered, we have $F_{i,[s+j]} = B_{i,[s+1]} + \sum_{l=1}^{j} p_{i,(l)}(s+l)^a$. For the first machine, the earliest starting time of nth job is the earliest completion time of (n−1)th job (i.e., $B_{1,[n]} = F_{1,[n-1]}$). In the context of Chung et al. [33] for unscheduled jobs, besides (s+1)th job on the second to the final machine, the procedure of computing the earliest starting time only considers the earliest completion time on the current machine and that immediately ahead of the machine (i.e., $E_{i,[s+j]} = \max\{F_{i,[s+j-1]}, F_{i-1,[s+j]}\}$). However, it may have the contradiction that the earliest starting time on the current machine is smaller than that on the preceding machines for the third and late machine. Therefore, to overcome the contradiction, we have

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

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

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

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

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

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

Step 5: If i=2, set $B_{i,[n]} = \max\{F_{i,[n-1]}, F_{i-1,[n]}\}$. Otherwise, set $B_{i,[n]} = \max\{F_{i,[n-1]}, F_{i-1,[n]}, B_{i-1,[n]}+ p_{i-1,(1)}\times n^a\}$.

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

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

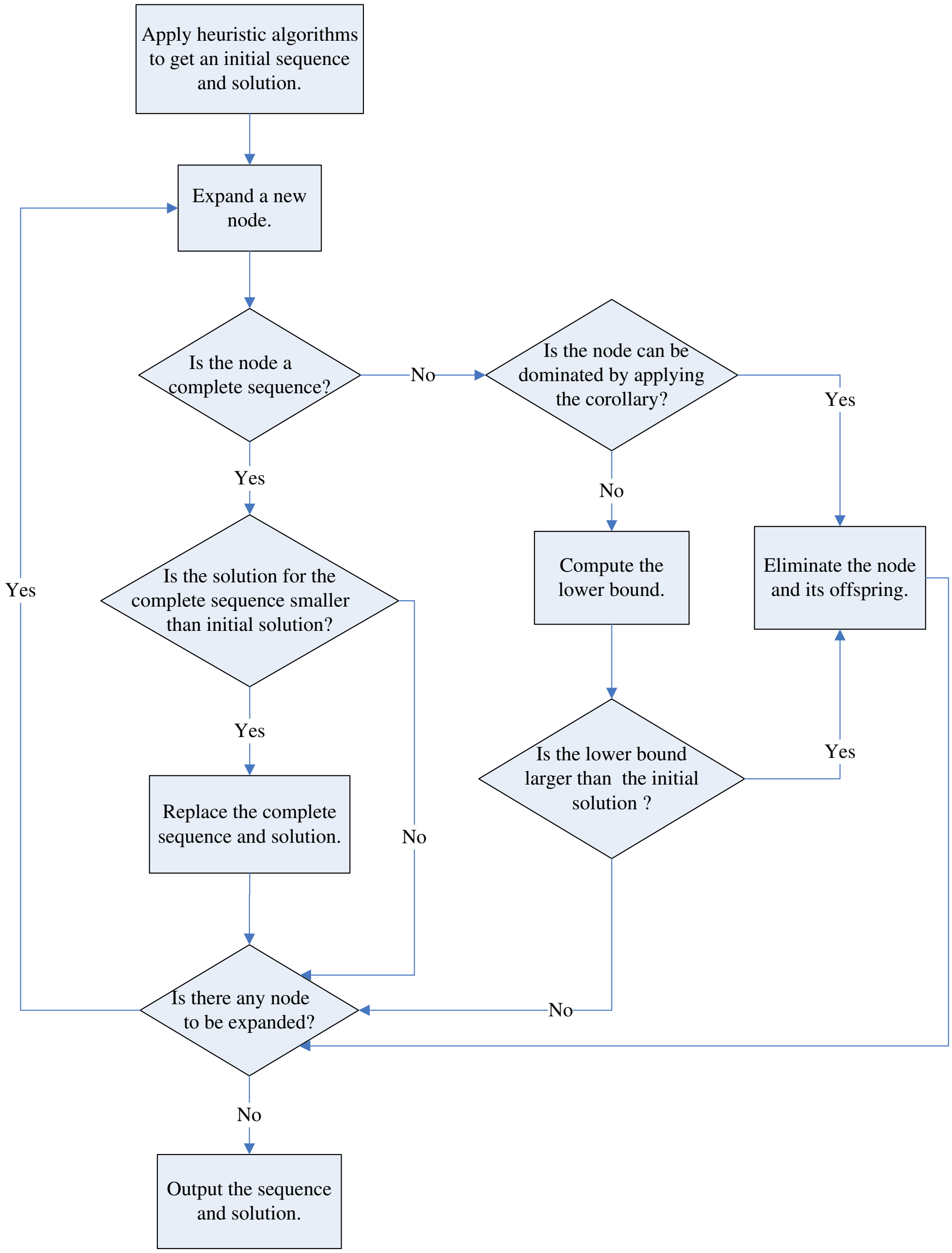
Step 8: The lower bound of the makespan for sequence σ is obtained as LB.

3.3 Heuristic algorithms

Seeking for the optimal sequence of a scheduling problem generally requires considerable computational time and memory for larger job-sized problems. Thus, this paper also focuses on assessing the performances of efficiency when applying economical heuristic algorithms with learning considerations to solve the scheduling problem.

The first algorithm is denoted as *NEH*. *NEH* is constructed by considering the learning effect to the algorithm proposed by Nawaz et al. [26]. The second algorithm is named as *RZ* in this paper. *RZ* modifies the algorithm which Rajendran and Ziegler [29] proposed by assuming the weights for all the jobs are equal, and we

Fig. 1 The flowchart of the proposed branch-and-bound algorithm



replace the total completion time by the makespan. The effect of learning is also considered in *RZ*. The third and final algorithms are denoted as *WY* and *FL*. *WY* and *FL*, respectively, modify the algorithm proposed by Woo and Yim [30] and Framinan and Leisten [31] by replacing the mean flow time by the makespan with the learning effect.

3.4 The procedure of the branch-and-bound algorithm

The branching procedure proposed in this paper adopts the depth-first search and assigns jobs in a forward manner starting from the first position. In the branching tree, the nodes

Table 1 The normal processing times for the demonstrated example

$p_{i,j}$		j Values			
		1	2	3	4
i	1	62	56	75	13
	2	18	30	4	100
	3	9	81	52	70

Table 2 The procedure to seek the optimal solution for the demonstrated example

Process	Node	Action	Reason
1	None	Apply heuristic algorithms to get an initial sequence and solution as (4,1,3,2) and 300.71	–
2	(1,–,–,–)	Eliminate the node	*LB*=304.75>300.71
3	(2,–,–,–)	None	–
4	(2,1,–,–)	None	–
5	(2,1,3,4)	Eliminate the node	Solution is 323.50>300.71
6	(2,1,4,3)	Eliminate the node	Solution is 313.98>300.71
7	(2,3,–,–)	None	–
8	(2,3,1,4)	Eliminate the node	Solution is 328.90>300.71
9	(2,3,4,1)	Replace (2,3,4,1) and 285.65 as the initial sequence and solution	Solution is 285.65<300.71
10	(2,4,–,–)	Eliminate the node	*LB*=288.74>285.65
11	(3,–,–,–)	None	–
12	(3,1,–,–)	Eliminate the node	*LB*=328.42>285.65
13	(3,2,–,–)	Eliminate the node	Dominated by (2,3,–,–)
14	(3,4,–,–)	Eliminate the node	Dominated by (4,3,–,–)
15	(4,–,–,–)	Eliminate the node	*LB*=300.71>285.65
16	None	Output sequence (2,3,4,1) and 285.65 as the optimal sequence and solution	No node can be expanded

are eliminated by the corollary or evaluating the lower bound. The detailed procedure is described as follows.

Step 1: Select the best schedule among the four heuristic algorithms as the initial solution.

Step 2: Expand the branching tree from node (–, –, …, –) to node (1, –, …, –), then to node (1, 2, –, …, –), and finally to node (n, n–1, …, 1) .

Step 3: Apply the corollary to check the node. If it is a dominated sequence, then eliminate the node.

Step 4: Evaluate the lower bound of the makespan for the current node or compute the makespan for the complete sequences. If the lower bound for the current node is larger than the initial solution, eliminate the node and all nodes beyond it in the branching tree. If the value of the complete sequence is smaller than the initial solution, then replace it as the new solution. Otherwise, eliminate it.

Step 5: Repeat Steps 2 to Step 4 until no more node can be expanded and the final initial solution is the optimal solution.

Furthermore, a flowchart is drawn in Fig. 1 to illustrate the detailed procedure of the branch-and-bound algorithm. Eventually, an illustrated example with four jobs and three machines is represented. The data are given in Table 1, and the steps are recorded in Table 2.

4 Computational results

We conduct a computational experiment in this section to assess the performance of the branch-and-bound algorithm and the four heuristic algorithms proposed in this paper. All the algorithms are coded in Fortran 90 and run on a Pentium 4 personal computer. The normal processing time

Table 3 The performance of the corollary and the lower bound for the branch-and-bound algorithm

m Value	a (%)	Number of mean nodes			Mean CPU times			
		B_C	*B_L*	*B_C+L*	*B_C*	*B_L*	*B_C+L*	Enumeration
3	90%	257236.9	917.7	450.4	4.234	0.031	0.017	15.504
	80%	183932.9	162.6	129.6	3.083	0.007	0.006	15.421
	70%	111829.0	92.7	78.2	1.949	0.005	0.004	15.379
5	90%	368537.7	945.1	771.2	10.067	0.067	0.056	25.148
	80%	250310.5	350.0	310.5	6.892	0.027	0.027	25.051
	70%	146816.5	134.3	122.3	4.031	0.012	0.012	24.806

Table 4 The performance of branch-and-bound algorithm and heuristic algorithms of different parameters

n Value	*m* Value	*a* (%)	Branch-and-bound algorithm								Heuristic algorithms							
			Number of nodes						CPU times		Error percentages (%)							
			Mean	SD	Q1	Q2	Q3	Number of outlier	Mean	SD	*NEH*		*RZ*		*WY*		*FL*	
											Mean	SD	Mean	SD	Mean	SD	Mean	SD
10	3	90%	450.4	1329.3	39	87	188	18	0.017	0.044	0.0134	0.0151	0.0375	0.0380	0.0167	0.0181	0.0062	0.0101
		80%	129.6	173.3	25	57	142	12	0.006	0.009	0.0193	0.0139	0.0320	0.0278	0.0211	0.0164	0.0064	0.0115
		70%	78.2	76.2	22	52	126	3	0.004	0.007	0.0359	0.0246	0.0299	0.0199	0.0236	0.0234	0.0069	0.0089
	5	90%	771.2	1647.3	98	190	760	11	0.056	0.106	0.0247	0.0206	0.0683	0.0429	0.0267	0.0260	0.0146	0.0156
		80%	310.5	548.6	59	135	275	10	0.027	0.039	0.0305	0.0202	0.0450	0.0324	0.0238	0.0223	0.0133	0.0143
		70%	122.3	161.9	26	62	152	7	0.012	0.015	0.0426	0.0308	0.0368	0.0268	0.0225	0.0205	0.0119	0.0118
12	3	90%	56719.8	384694.2	78	288	1083	20	2.204	14.296	0.0135	0.0116	0.0405	0.0337	0.0191	0.0191	0.0066	0.0098
		80%	1192.9	3480.4	86	299	612	15	0.065	0.169	0.0241	0.0172	0.0354	0.0323	0.0223	0.0194	0.0068	0.0079
		70%	521.4	849.3	71	255	550	11	0.033	0.045	0.0411	0.0244	0.0377	0.0238	0.0257	0.0189	0.0080	0.0075
	5	90%	9448.4	34991.5	379	1093	4818	12	0.884	3.174	0.0253	0.0195	0.0674	0.0418	0.0276	0.0208	0.0146	0.0129
		80%	2772.7	9635.3	158	510	1722	12	0.275	0.837	0.0336	0.0196	0.0541	0.0373	0.0274	0.0238	0.0152	0.0144
		70%	583.2	1109.3	64	188	630	11	0.064	0.103	0.0513	0.0249	0.0444	0.0276	0.0235	0.0190	0.0105	0.0122
14	3	90%	167322.1	854106.4	200	1442	5900	14	8.630	44.147	0.0134	0.0098	0.0405	0.0278	0.0149	0.0146	0.0060	0.0107
		80%	19161.7	124696.6	295	988	5848	6	1.082	6.403	0.0280	0.0145	0.0433	0.0315	0.0231	0.0170	0.0080	0.0094
		70%	1720.3	2910.8	131	509	1623	12	0.148	0.234	0.0497	0.0224	0.0411	0.0213	0.0256	0.0192	0.0073	0.0072
	5	90%	301967.5	1838143.9	1583	4289	15460	20	27.966	151.640	0.0285	0.0170	0.0845	0.0473	0.0341	0.0264	0.0163	0.0142
		80%	9213.4	22874.2	622	2016	5053	19	1.263	2.934	0.0352	0.0194	0.0484	0.0311	0.0274	0.0200	0.0137	0.0110
		70%	6369.0	33345.7	486	1370	3463	11	0.867	3.795	0.0531	0.0250	0.0479	0.0238	0.0267	0.0170	0.0135	0.0121
16	3	90%	2111749.8	11723845.2	659	3214	26519	17	125.597	685.231	0.0140	0.0090	0.0404	0.0310	0.0185	0.0156	0.0051	0.0069
		80%	41433.3	148176.9	900	2762	17081	12	3.367	10.539	0.0207	0.0188	0.0416	0.0238	0.0237	0.0161	0.0079	0.0081
		70%	22073.5	74962.3	406	2102	10563	16	2.031	5.763	0.0485	0.0235	0.0470	0.0247	0.0253	0.0179	0.0084	0.0101
	5	90%	1055484.8	4641812.1	6442	14074	94299	15	116.626	462.804	0.0253	0.0170	0.0816	0.0452	0.0272	0.0180	0.0142	0.0133
		80%	123731.8	447437.6	2384	9808	30383	18	19.762	67.893	0.0396	0.0200	0.0627	0.0382	0.0315	0.0181	0.0152	0.0120
		70%	19159.7	72050.8	1136	3873	12001	11	3.190	8.667	0.0524	0.0237	0.0564	0.0253	0.0287	0.0210	0.0120	0.0107
18	3	90%	8470804.1	26263090.6	8173	46669	335005	17	451.124	1358.185	0.0144	0.0093	0.0411	0.0280	0.0173	0.0147	0.0074	0.0094
		80%	593669.8	3611399.3	2219	16609	86473	11	45.948	251.540	0.0308	0.0166	0.0429	0.0303	0.0238	0.0157	0.0090	0.0102
		70%	75422.6	211051.6	1753	11367	48150	13	7.565	18.756	0.0493	0.0202	0.0447	0.0213	0.0300	0.0171	0.0088	0.0081
	5	90%	9241667.3	24428652.2	40402	172912	2336244	19	1089.132	2811.171	0.0252	0.0139	0.0813	0.0478	0.0313	0.0196	0.0176	0.0142
		80%	449812.6	1760902.3	8801	48120	141615	13	64.596	222.659	0.0430	0.0182	0.0674	0.0350	0.0330	0.0212	0.0143	0.0131
		70%	97797.9	339614.2	3177	10124	56667	15	17.122	46.489	0.0555	0.0216	0.0616	0.0246	0.0286	0.0178	0.0139	0.0097

of all jobs are generated from a discrete uniform distribution over 1 to 100.

4.1 Performance of the algorithms for small job-sized problems

In order to test the efficiency of the proposed corollary and the lower bound, a computational experiment is implemented with fixed job size at 10, two different machine sizes at 3 and 5, 100 replications, and three different levels of learning effects at 90%, 80%, and 70% (which correspond to $a=-0.152$, $a=-0.322$, and $a=-0.515$.). The results are listed in Table 3, in which *B_C* denotes the branch-and-bound algorithm with only the corollary, *B_L* denotes the branch-and-bound algorithm with only the lower bound, and *B_C+L* denotes the branch-and-bound algorithm with both the corollary and the lower bound. In addition, the mean number of nodes and the mean execution time are recorded. Meanwhile, the mean execution time for the enumeration method is also recorded. As shown in Table 3, the efficiency of the corollary and the lower bound in the branch-and-bound algorithm are significant in terms of the mean execution time by comparison with the enumeration method. Furthermore, the lower bound is more effective than the corollary in terms of the mean number of nodes and the mean execution time, and the phenomenon is notable when the learning effect is stronger. However, the most efficient performance is exhibited when *B_C+L* is implemented in terms of the mean number of nodes and the mean execution time. Therefore, the branch-and-bound algorithm with both the corollary and the lower bounds is recommended for the succeeding computational experiment in this paper.

We use five job sizes ($n=10$, 12, 14, 16, and 18) and two different machine sizes ($m=3$ and 5) to yield the optimal solution and test the accuracy of all the proposed heuristic algorithms. Furthermore, to examine the influence of learning effects, the learning effects are taken to be 90%, 80%, and 70%. Consequently, 30 experimental conditions are examined, and 100 replications are randomly generated for each condition. A total of 3,000 instances are generated, and the results are listed in Table 4. The mean and the standard deviation of the number of nodes and of the execution time for the proposed branch-and-bound algorithm are recorded. In addition, the mean and standard deviation of the error percentages for the four heuristic algorithms are also recorded. For each instance, the error percentage of the given heuristic algorithm is calculated as

$$\left(V - V^*\right)\Big/V^* \times 100\%,$$

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

It is observed that the four heuristic algorithms proposed in this paper are quite accurate since all the mean error percentages are less than 0.1%. Furthermore, *FL* has the best performance and *RZ* has the worst performance. From the results of the branch-and-bound algorithm, it reveals that, for the problem proposed in this paper, it is easier to obtain the optimal solution in terms of the mean number of nodes when the learning effect strengthens. However, the standard deviation of the number of nodes exceeds its mean for all the cases, which implies that there are worst cases with a tremendous number of nodes. Therefore, the quartile of 25%, 50%, and 75% for the number of nodes is evaluated and recorded as Q1, Q2, and Q3. The observations show that the distribution for the number of nodes is right skewed because most of the mean numbers of nodes are relatively large to Q3, and it implies that most of the instances have fewer nodes. For the same instances, the box-plot of logarithm scale for the number of nodes with different parameters for the learning effect as 90%, 80%, and 70% is shown in Figs. 2, 3, and 4, respectively. The figures illustrate that the number of nodes and the execution time grow exponentially with an increasing number of jobs.

In order to investigate the influence of outliers, the number of outliers for each experimental condition is listed in Table 1, where the number of nodes for given instance which exceeds the value of Q3+1.5(Q3−Q1) is recorded as the outlier. The outliers are eliminated, and the performance of the branch-and-bound algorithm is shown in Table 5.

Table 5 illustrates that the means and the standard deviations for the number of nodes and execution time are all reduced by a wide margin after eliminating the outliers. Eventually, since the quantity of outliers is less than 20% of all instances for each experimental condition in this paper, we recommend to conduct the proposed branch-and-bound algorithm for obtaining the optimal solution within a reasonable amount of time, or conduct the proposed heuristic algorithms for obtaining near-optimal solutions when the number of jobs is larger than 18.

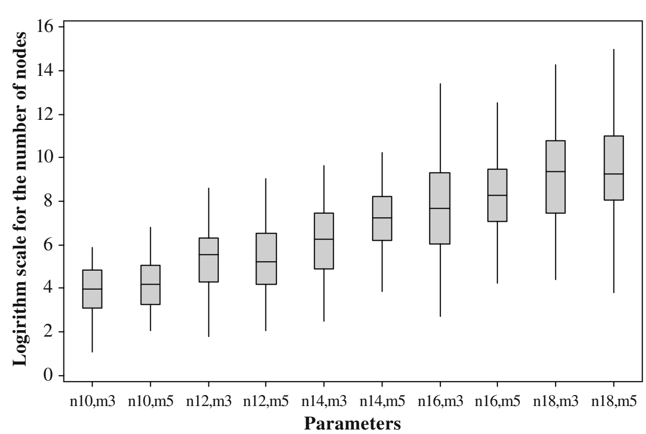


Fig. 2 Box-plot for logarithm scale with learning effect as 70%

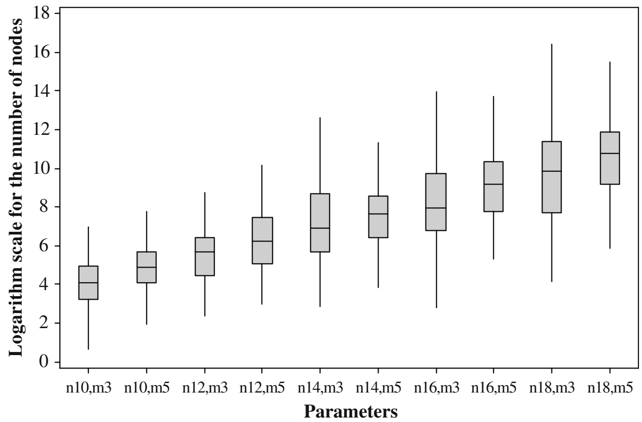


Fig. 3 Box-plot for logarithm scale with learning effect as 80%

4.2 Performance of the algorithms for large job-sized problems

To indicate the performance of the proposed heuristic algorithms for large job-sized problems with learning considerations, we use three different job sizes (n=50, 100, and 150), four different machine sizes (m=5, 10, 15, and 20) and three learning effects (90%, 80%, and 70%) to yield the near-optimal solutions. The mean and the standard deviation of relative percentage deviation (*RPD*) are reported for each heuristic algorithm. For each instance, the *RPD* is obtained with respect to the best one of all near-optimal solutions generated by the four heuristic algorithms, i.e., $RPD=V/V_{\min}$, where V denotes the value of the makespan generated by the given heuristic algorithm and $V_{\min}$ denotes the minimal one among the values of the makespan generated by the four heuristic algorithms. Consequently, 36 experimental conditions are examined, and 100 replications are randomly generated for each condition. A total of 3,600 instances are generated, and the results are listed in Table 6.

In Table 6, the value of *RPD* from *FL* is the minimal one among four heuristic algorithms for every experiment condition. The observation shows that *FL* is more accurate than the other three heuristic algorithms. However, as all the *RPD* values are greater than 1, there is no algorithm which completely dominates the others. From the values of *RPD* for the four heuristic algorithms, one-way analysis of variance (ANOVA) with a significance of 5% is applied to test that the mean values of *RPD* are all the same among four algorithms or whether at least one differs from the others. The results are given in Table 7.

Since the p value is below the significance level, it implies that the mean values of *RPD* are not all identical. Therefore, the efficiency among the four heuristic algo-

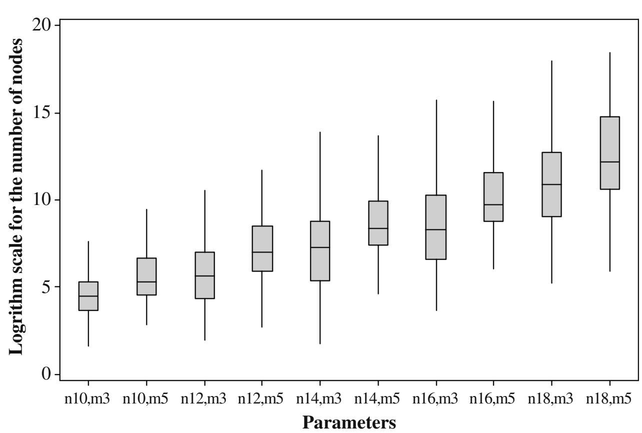


Fig. 4 Box-plot for logarithm scale with learning effect as 90%

Table 5 The performance of branch-and-bound algorithm of different parameters after outliers elimination

n Value	m Value	a (%)	Branch-and-bound algorithm			
			Number of nodes		CPU times	
			Mean	SD	Mean	SD
10	3	90%	89.7	80.5	0.005	0.007
		80%	78.5	75.1	0.004	0.007
		70%	70.8	64.2	0.003	0.007
	5	90%	337.5	365.4	0.028	0.026
		80%	164.0	143.5	0.016	0.014
		70%	85.8	75.5	0.009	0.010
12	3	90%	355.9	435.2	0.022	0.026
		80%	307.0	298.7	0.021	0.022
		70%	268.7	252.9	0.019	0.018
	5	90%	1912.0	2334.1	0.204	0.237
		80%	761.0	858.6	0.090	0.089
		70%	287.0	317.0	0.036	0.040
14	3	90%	2431.5	3104.2	0.195	0.234
		80%	2605.9	3474.3	0.210	0.272
		70%	801.0	964.7	0.075	0.084
	5	90%	5655.6	6874.0	0.853	0.946
		80%	2009.5	2020.1	0.328	0.307
		70%	1800.1	1840.8	0.317	0.319
16	3	90%	7586.8	10851.7	0.771	1.032
		80%	6814.2	9770.4	0.712	0.981
		70%	3646.5	5035.8	0.426	0.568
	5	90%	33524.6	49524.8	6.176	8.746
		80%	10837.8	12194.7	2.415	2.769
		70%	5519.6	6149.3	1.198	1.251
18	3	90%	115505.8	174775.7	11.263	16.210
		80%	36878.3	51417.1	4.025	5.316
		70%	18440.8	23462.5	2.079	2.605
	5	90%	566117.6	1071722.1	82.906	144.164
		80%	67915.4	82120.5	14.043	16.320
		70%	20391.9	28773.9	4.521	5.776

Table 6 The relative percentage deviation of heuristic algorithms

n Value	*m* Value	*a* (%)	Relative percentage deviation (*RPD*)							
			NEH		*RZ*		*WY*		*FL*	
			Mean	SD	Mean	SD	Mean	SD	Mean	SD
50	5	90%	1.0142	0.0074	1.0479	0.0268	1.0133	0.0100	1.0009	0.0029
		80%	1.0379	0.0130	1.0493	0.0167	1.0172	0.0117	1.0000	0.0004
		70%	1.0654	0.0194	1.0720	0.0254	1.0195	0.0166	1.0005	0.0023
	10	90%	1.0179	0.0113	1.0877	0.0322	1.0138	0.0109	1.0010	0.0027
		80%	1.0413	0.0159	1.0677	0.0279	1.0151	0.0111	1.0003	0.0015
		70%	1.0610	0.0250	1.0787	0.0235	1.0126	0.0112	1.0010	0.0038
	15	90%	1.0185	0.0130	1.0975	0.0243	1.0161	0.0122	1.0012	0.0034
		80%	1.0429	0.0188	1.0694	0.0247	1.0130	0.0096	1.0006	0.0022
		70%	1.0584	0.0171	1.0766	0.0217	1.0117	0.0112	1.0010	0.0028
	20	90%	1.0204	0.0145	1.1013	0.0243	1.0182	0.0121	1.0006	0.0020
		80%	1.0432	0.0184	1.0689	0.0212	1.0125	0.0112	1.0003	0.0015
		70%	1.0587	0.0200	1.0727	0.0203	1.0079	0.0086	1.0007	0.0020
100	5	90%	1.0175	0.0052	1.0350	0.0159	1.0106	0.0067	1.0004	0.0016
		80%	1.0437	0.0107	1.0524	0.0175	1.0144	0.0091	1.0001	0.0012
		70%	1.0747	0.0178	1.0832	0.0234	1.0184	0.0115	1.0001	0.0012
	10	90%	1.0165	0.0079	1.0750	0.0243	1.0127	0.0082	1.0003	0.0011
		80%	1.0466	0.0128	1.0662	0.0204	1.0158	0.0083	1.0000	0.0003
		70%	1.0731	0.0197	1.0932	0.0193	1.0135	0.0098	1.0003	0.0016
	15	90%	1.0182	0.0090	1.0956	0.0266	1.0122	0.0086	1.0001	0.0008
		80%	1.0481	0.0166	1.0734	0.0210	1.0132	0.0082	1.0002	0.0011
		70%	1.0665	0.0163	1.0916	0.0179	1.0117	0.0091	1.0004	0.0015
	20	90%	1.0183	0.0097	1.1036	0.0200	1.0117	0.0083	1.0003	0.0010
		80%	1.0501	0.0154	1.0760	0.0212	1.0121	0.0077	1.0001	0.0010
		70%	1.0655	0.0193	1.0871	0.0201	1.0068	0.0076	1.0017	0.0046
150	5	90%	1.0197	0.0052	1.0291	0.0132	1.0082	0.0058	1.0005	0.0024
		80%	1.0477	0.0090	1.0448	0.0136	1.0122	0.0084	1.0006	0.0030
		70%	1.0774	0.0155	1.0894	0.0252	1.0164	0.0086	1.0001	0.0005
	10	90%	1.0180	0.0065	1.0655	0.0200	1.0101	0.0065	1.0002	0.0007
		80%	1.0497	0.0119	1.0653	0.0175	1.0121	0.0061	1.0001	0.0009
		70%	1.0782	0.0175	1.1006	0.0162	1.0141	0.0117	1.0002	0.0011
	15	90%	1.0187	0.0064	1.0902	0.0208	1.0098	0.0063	1.0001	0.0010
		80%	1.0495	0.0147	1.0721	0.0180	1.0119	0.0065	1.0001	0.0006
		70%	1.0698	0.0189	1.0978	0.0156	1.0113	0.0082	1.0006	0.0017
	20	90%	1.0180	0.0068	1.0992	0.0195	1.0098	0.0062	1.0002	0.0009
		80%	1.0515	0.0142	1.0739	0.0177	1.0116	0.0067	1.0000	0.0004
		70%	1.0696	0.0200	1.0940	0.0175	1.0101	0.0082	1.0002	0.0007

rithms should be considered. Furthermore, the Tukey's test with a significance of 5% is implemented to compare the values of *RPD* among the four heuristic algorithms. The results of Tukey's test are summarized in Table 8.

The test results imply that *FL* is the best among the four algorithms, follows by *WY* and *NEH*, and finally *RZ*. Thus, the algorithm adapted from Framinan and Leisten [31] is recommended to obtain the near-optimal solution for the

Table 7 One-way ANOVA for *RPD* of four heuristics

Source	*DF*	*SS*	*MS*	*F*	*p* Value
Factor	3	0.124511	0.041504	199.66	0.000
Error	140	0.029101	0.000208		
Total	143	0.153612			

Table 8 Tukey's test results of four heuristics

	Lower	Center	Upper
NEH subtracted from			
RZ	0.02331	0.03215	0.04100
WY	−0.04009	−0.03124	−0.02240
FL	−0.05249	−0.04365	−0.03481
RZ subtracted from			
WY	−0.07224	−0.06340	−0.05455
FL	−0.08465	−0.07580	−0.06696
WY subtracted from			
FL	−0.02125	−0.01241	−0.00356

makespan problem with learning considerations in flowshop setting.

5 Conclusion

This paper examines an m-machine permutation flowshop problem with learning considerations where the aim is to minimize the makespan. A dominance theorem and a lower bound are proposed to conduct a branch-and-bound procedure for optimizing the solution. In addition, this paper also introduces learning effects to four well-known existing heuristic algorithms and adapts them to solve the scheduling problem. The computational results show that the branch-and-bound algorithm can solve problems of up to 18 jobs within a reasonable amount of time and demonstrate that *FL* performs best for small job-sized problems. Meanwhile, for large job-sized problems, *FL* also has identical performance. Therefore, we recommend the heuristic algorithm adapted from Framinan and Leisten [31] to obtain the approximate solution. Eventually, since the heuristic algorithms for the position-based learning proposed in this paper are not affected by different learning index, the discussion for sum-of-processing-time-based learning is attractive in future research.

References

1. Pinedo M (2002) Scheduling: theory, algorithms, and systems. Prentice-Hall, Upper Saddle River
2. Smith WE (1956) Various optimizers for single state production. Nav Res Logist Q 3:59–66
3. Biskup D (1999) Single-machine scheduling with learning considerations. Eur J Oper Res 115:173–178
4. Wright TP (1936) Factors affecting the cost of airplanes. J Aeronaut Sci 3:122–128
5. Yelle LE (1979) The learning curve: historical review and comprehensive survey. Decis Sci 10:302–328
6. Cheng TCE, Wu CC, Lee WC (2008) Some scheduling problems with sum-of-processing-times-based and job-position-based learning effects. Inf Sci 178:2476–2487
7. Biskup D (2008) A state-of-the-art review on scheduling with learning effect. Eur J Oper Res 188:315–329
8. Wang XR, Wang JB, Gao WJ, Huang X (2010) Scheduling with pasr-sequence-dependent setup times and learning effects on single machine. Int J Adv Manuf Technol 48:739–746
9. Wang JB, Sun L, Sun L (2010) Single machine scheduling with a learning effect a discounted costs. Int J Adv Manuf Technol 49:1141–1149
10. Janiak A, Rudek R (2008) A new approach to the learning effect: beyond the learning curve restrictions. Comput Oper Res 35:3727–3736
11. Janiak A, Rudek R (2009) Experience based approach to scheduling problems with the learning effect. IEEE Trans Syst Man Cybern, Part A, Syst Humans 39:344–357
12. Toksari MD, Güner E (2009) Parallel machine earliness/tardiness scheduling problem under the effects of position based learning and linear/nonlinear deterioration. Comput Oper Res 36:2394–2417
13. Eren T, Güner E (2009) A bicriteria parallel machine scheduling with a learning effect. Int J Adv Manuf Technol 40:1202–1205
14. Koulamas C, Kyparisis GJ (2007) Single-machine and two-machine flowshop scheduling with general learning function. Eur J Oper Res 178:402–407
15. Wu CC, Hsu PH, Chen JC, Wang NS, Wu WH (2010) Branch-and-bound and simulated annealing algorithms for a total weighted completion time scheduling with ready time and learning effect. Int J Adv Manuf Technol. doi:10.1007/s00170-010-3022-7
16. Cheng TCE, Lai PJ, Wu CC, Lee WC (2009) Single-machine scheduling with sum-of-logarithm-processing-times-based learning considerations. Inf Sci 179:3127–3135
17. Wang LY, Wang JB, Wang D, Yin N, Huang X, Feng EM (2009) Single-machine scheduling with a sum-of-processing-time based learning effect and deteriorating jobs. Int J Adv Manuf Technol 45:336–340
18. Wang LY, Wang JB, Gao WJ, Huang X, Feng EM (2010) Two single-machine scheduling problems with the effects of deterioration and learning. Int J Adv Manuf Technol 46:715–720
19. Yin Y, Xu D, Sun K, Li H (2009) Some scheduling problems with general position-dependent and time-dependent learning effects. Inf Sci 179:2416–2425
20. Lee WC, Wu CC (2009) Some single-machine and m-machine flowshop scheduling problems with learning considerations. Inf Sci 179:3885–3892
21. Wu CC, Lee WC, Wang WC (2007) A two-machine flowshop maximum tardiness scheduling problem with a learning effect. Int J Adv Manuf Technol 31:743–750
22. Lee WC, Wu CC (2004) Minimizing total completion time in a two-machine flowshop with a learning effect. Int J Prod Econ 88:85–93
23. Chen P, Wu CC, Lee WC (2006) A bi-criteria two-machine flowshop scheduling problem with a learning effect. J Oper Res Soc 57:1113–1125
24. Wang JB, Xia ZQ (2005) Flow-shop scheduling with a learning effect. J Oper Res Soc 56:1325–1330
25. Wu CC, Lee WC (2009) A note on the total completion time problem in a permutation flowshop with a learning effect. Eur J Oper Res 192:343–347
26. Nawaz M, Enscore EE, Ham I (1983) A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. OMEGA 11:91–95

27. Liu S, Ong HL (2002) A comparative study of algorithms for the flowshop scheduling problem. Asia-Pac J Oper Res 19:205–222
28. Ruiz R, Maroto C (2005) A comprehensive review and evaluation of permutation flowshop heuristics. Eur J Oper Res 165:479–494
29. Rajendran C, Ziegler H (1997) An efficient heuristic for scheduling in a flowshop to minimize total weighted flowtime of jobs. Eur J Oper Res 103:129–138
30. Woo HS, Yim DS (1998) A heuristic algorithm for mean flowtime objective in flowshop scheduling. Comput Oper Res 25:175–182
31. Framinan JM, Leisten R (2003) An efficient constructive heuristic for flowtime minimization in permutation flow shops. OMEGA 31:311–317
32. Garey MR, Johnson DS, Sethi R (1976) The complexity of flowshop and jobshop scheduling. Math Oper Res 1:117–129
33. Chung CS, Flynn J, Kirca Ő (2002) A branch-and-bound algorithm to minimize the total flow time for m-machine permutation flowshop problems. Int J Prod Econ 79:185–196
34. Chung CS, Flynn J, Kirca Ő (2006) A branch and bound algorithm to minimize the total tardiness for *m*-machine permutation flowshop problems. Eur J Oper Res 174:1–10