Hindawi
Complexity
Volume 2018, Article ID 9149510, 12 pages
https://doi.org/10.1155/2018/9149510

Research Article

Bicriterion Optimization for Flow Shop with a Learning Effect Subject to Release Dates

Ji-Bo Wang,[1] Jian Xu,[2] and Jing Yang[2]

[1]*School of Science, Shenyang Aerospace University, Shenyang 110136, China*
[2]*School of Management Science and Engineering, Dongbei University of Finance and Economics, Dalian 116025, China*

Correspondence should be addressed to Jian Xu; xujian@dufe.edu.cn

Received 30 March 2018; Accepted 15 July 2018; Published 17 October 2018

Academic Editor: Michele Scarpiniti

This paper investigates a two-machine flow shop problem with release dates in which the job processing times are variable according to a learning effect. The bicriterion is to minimize the weighted sum of makespan and total completion time subject to release dates. We develop a branch-and-bound (B&B) algorithm to solve the problem by using a dominance property, several lower bounds, and an upper bound to speed up the elimination process of the search tree. We further propose a multiobjective memetic algorithm (MOMA), enhanced by an initialization strategy and a global search strategy, to obtain the Pareto front of the problem. Computational experiments are also carried out to examine the effectiveness and the efficiency of the B&B algorithm and the MOMA algorithm.

1. Introduction

In many industrial systems, logistics, and service settings, the processing times of jobs can decrease due to the learning effects, i.e., firms and employees perform a job (task) over and over and they learn how to perform more efficiently [1, 2]. This is especially observed in *seru* production system, consisting of some equipment and workers who produce one or more part types. After undertaking cross-training, the learning effects greatly affect their efficiencies [3]. "As there is a significant involvement of humans in scheduling environments, the number of activities subject to learning is high, too. Hence it seems reasonable to consider learning in scheduling environments" [2]. On the other hand, the importance of flow shop scheduling optimization is widely recognized in many manufacturing and assembling processes, e.g., the internet connectivity problem in 3-tiered client-server databases is a two-machine flow shop problem (see [4]); the scheduling of multimedia date objectives for WWW applications reduces to a two-machine flow shop problem (see [5–7]; see also the reviews given in [6] and [8–10]).

More recently, Xu et al. [11], Eren [12], Wu and Lee [13], Rudek [14], Kuo et al. [15], Lee and Chung [16], Sun et al. [17, 18], Cheng et al. [19], Li et al. [20], Wang and Zhang [21], J. B. Wang and J. J. Wang [22], Lai et al. [23], Liu and Feng [24], Wang and Zhang [21], Wu et al. [25], Shiau et al. [26], Lu [27], Qin et al. [28], He [29], and Wang et al. [30] considered flow shop scheduling problems with learning effects, but without release dates. Xu et al. [11] considered the flow shop scheduling problems with position-dependent learning effect, i.e., if job J_j is in position l of a schedule, the actual processing time p_{ijl} of job J_j on machine M_i is $p_{ijl} = p_{ij}(a - bl)$ and $p_{ijl} = p_{ij}l^{\alpha}$, where p_{ij} denotes the normal processing time of job J_j on M_i and $a > 0$, $b \geq 0$, and $\alpha \leq 0$ are the learning rates. For several regular objective functions, they presented approximate algorithms. Cheng et al. [19] considered the model $p_{ijl} = p_{ij}l^{\alpha}$. For the maximum lateness minimization, they gave a mathematical programming model. Wang et al. [31] considered the model $p_{ijl} = p_{ij}l^{\alpha}$. For the total completion time minimization, they gave a branch-and-bound algorithm and several well-known

heuristics. Lai et al. [23] considered the model $p_{ijl} = p_{ij}l^{\alpha}$. For the total tardiness minimization, they gave a branch-and-bound algorithm and two heuristics. He [29] and Li et al. [20] considered the flow shop scheduling problems with time-dependent learning effect, i.e., if job J_j is in position l of a schedule, the actual processing time p_{ijl} of job J_j on machine M_i is $p_{ijl} = p_{ij}(1 + \sum_{h=1}^{l-1} p_{i[h]})^{a}$, where $a \le 0$ is the learning rate and $[h]$ denotes the job that occupies the hth position in a sequence. For several regular objective functions, they presented approximate algorithms. Shiau et al. [26] considered the flow shop scheduling problems with general position-dependent learning effect, i.e., if job J_j is in position l of a schedule, the actual processing time p_{ijl} of job J_j on machine M_i is $p_{ijl} = p_{ij}g(l)$, where $1 = g(1) \ge g(2) \le \cdots \ge g(n)$ is a nonincreasing function. Sun et al. [18] considered the total weighted completion time minimization flow shop scheduling problem. For three position-dependent learning effects, they gave heuristic algorithms. Cheng et al. [19], Lai et al. [23], Wang et al. [31], Wu and Lee [13], and Wang et al. [30] considered the flow shop scheduling with truncated learning effects. J.-B. Wang and J.-J. Wang [22] considered the flow shop scheduling with a general exponential learning effect. For five regular objective functions, they presented heuristic algorithms. Wang and Zhang [21] considered the flow shop scheduling with position-weighted learning effects. For the weighted sum of makespan and total completion time minimization, they gave a branch-and-bound algorithm and some heuristic algorithms. Qin et al. [28] considered the flow shop group scheduling with position-based learning effect. For four objectives (i.e., the makespan, total completion time, total weighted completion time, and maximum lateness), they gave several heuristics and metaheuristics. For new trends in flow shop scheduling with learning effects, we refer the reader to Rudek [14], Liu and Feng [24], Shiau et al. [26], Lu [27], and He [29].

However, in fact, the scheduling with release dates is interesting and closer to real problems [32–34]. With modeling a realistic production system in mind, Yin et al. [35] considered single-machine scheduling with release dates and position-dependent learning effects. To the best of our knowledge, the flow shop scheduling problems with a learning effect and release dates (i.e., the ready times) simultaneously have barely been investigated. Bai et al. [36] is the only identifiable exception. Hence, in this paper, we consider the two-machine flow shop scheduling problem with position-dependent learning effect and release dates, and the objective is to minimize the weighted sum of makespan and total completion time. Obviously, the problem under study is NP-hard [34]; thus, the branch-and-bound algorithm might be a good way to obtain the optimal schedule. We further design an evolutionary multiobjective optimization algorithm to obtain the Pareto front of high quality. This contribution makes a clear distinction between this paper and Bai et al. [36], which solely focuses on a single aggregated objective.

The remaining of the paper is organized as follows. In Section 2, we formalize the problem. Dominance conditions for the problem are presented in Section 3. Branch-and-bound algorithm is discussed in Section 4. Numerical experiments are conducted in Section 5. The last section concludes the paper.

2. Notations and Assumptions

The problem may be stated as follows. We are given n jobs and two machines M_1 and M_2. Each machine is available at time zero and the job J_j becomes available at its release date $r_j \ge 0$. Let $\mathcal{J} = \{J_1, \ldots, J_j, \ldots, J_n\}$ represent the set of jobs which are to be processed on 2 machine permutation flow shop settings. Each job J_j $(j = 1, 2, \ldots, n)$ is required to be processed on machine M_1 and then M_2. As in Biskup [2], Lee and Chung [16], Lee and Wu [37], and Lee et al. [38], if job J_j is in position l of a schedule, then the actual processing times a_{jl} and b_{jl} of job J_j on machines M_1 and M_2 are

$$\begin{aligned} a_{jl} &= a_j l^{\alpha}, \quad l, j = 1, 2, \ldots, n, \\ b_{jl} &= b_j l^{\alpha}, \quad l, j = 1, 2, \ldots, n, \end{aligned} \tag{1}$$

respectively, where $a_j(b_j)$ denotes the normal processing time (i.e., the basic processing time before learning effects happened) of job J_j on $M_1(M_2)$ and $\alpha = \log_2 \mathrm{LR} \le 0$ is the learning index for both machines M_1 and M_2 depending on the learning rate LR. For a given schedule π, let $C_{1j}(\pi) = C_{1j}(C_{2j}(\pi) = C_j)$ be the completion time of job J_j on machine $M_1(M_2)$. The objective is to minimize $\lambda C_{\max} + (1 - \lambda)\sum_{j=1}^{n} C_j$, where $C_{\max} = \max\{C_j|_{j=1,2,\ldots,n}\}$ is the makespan and $\sum_{j=1}^{n} C_j$ is the total completion time, $0 \le \lambda \le 1$. Using the conventional three-field notation for describing scheduling problems (Graham et al. [39]), the problem under consideration can be described as $F2|\mathrm{LE}, r_j|\lambda C_{\max} + (1 - \lambda)\sum_{j=1}^{n} C_j$, where LE denotes the learning effect [1, 2].

3. Dominance Conditions

In this section, we give some dominance rules for $F2|\mathrm{LE}, r_j|\lambda C_{\max} + (1 - \lambda)\sum_{j=1}^{n} C_j$. "Dominance rules are basically necessary conditions for any optimal schedule that can be generated. Applying such rules results in a set of precedence relations between jobs. These precedence relations are then used to reduce the number of branches in a branch-and-bound search tree" (Yin et al. [35]). Let $S_1 = (\pi, J_j, J_k, \pi')$ and $S_2 = (\pi, J_k, J_j, \pi')$ be obtained from S_1 by only interchanging jobs J_j and J_k, where π and π' are partial sequences, and there are $l - 1$ jobs in π. To show that S_2 dominates S_1, it is sufficient to show that $C_{1j}(S_2) \le C_{1k}(S_1)$, $C_j(S_2) \le C_k(S_1)$, and $C_k(S_2) + C_j(S_2) \le C_k(S_1) + C_j(S_1)$. Let $A(B)$ denote the last completion time prior to jobs J_j and J_k on machine M_1 (M_2) in S_1; obviously, A and B remain unchanged in S_2.

Proposition 1. *If the jobs J_k and J_j satisfy the following conditions:*

(i) Either $r_k + a_k l^\alpha + b_k l^\alpha \le r_j + a_j l^\alpha + b_j l^\alpha$ or $r_k + a_k l^\alpha + b_k l^\alpha \le A + a_j l^\alpha + b_j l^\alpha$ or $r_k + a_k l^\alpha + b_k l^\alpha \le B + b_j l^\alpha$

(ii) Either $A + a_k l^\alpha + b_k l^\alpha \le r_j + a_j l^\alpha + b_j l^\alpha$ or $a_k + b_k \le a_j + b_j$ or $A + a_k l^\alpha + b_k l^\alpha \le B + b_j l^\alpha$

(iii) Either $B + b_k l^\alpha \le r_j + a_j l^\alpha + b_j l^\alpha$ or $B + b_k l^\alpha \le A + a_j l^\alpha + b_j l^\alpha$ or $b_k \le b_j$

(iv) Either $r_k + a_k l^\alpha + a_j (l+1)^\alpha \le r_j + a_j l^\alpha + a_k (l+1)^\alpha$ or $A + a_k l^\alpha + a_j (l+1)^\alpha \le r_j + a_j l^\alpha + a_k (l+1)^\alpha$;

(v) Either $r_k + a_k l^\alpha + a_j (l+1)^\alpha \le A + a_j l^\alpha + a_k (l+1)^\alpha$ or $a_k l^\alpha + a_j (l+1)^\alpha \le a_j l^\alpha + a_k (l+1)^\alpha$ or $r_j + a_j (l+1)^\alpha \le A + a_j l^\alpha + a_k (l+1)^\alpha$

(vi) Either $A + a_k l^\alpha + a_j (l+1)^\alpha \le r_k + a_k (l+1)^\alpha$ or $r_j + a_j (l+1)^\alpha \le r_k + a_k (l+1)^\alpha$

(vii) Either $a_k l^\alpha + a_j (l+1)^\alpha + b_j (l+1)^\alpha \le a_j l^\alpha + a_k (l+1)^\alpha + b_k (l+1)^\alpha$ or $a_k l^\alpha + a_j (l+1)^\alpha + b_j (l+1)^\alpha \le a_j l^\alpha + b_j l^\alpha + b_k (l+1)^\alpha$ or $A + a_k l^\alpha + a_j (l+1)^\alpha + b_j (l+1)^\alpha \le B + b_j l^\alpha + b_k (l+1)^\alpha$ or $A + a_k l^\alpha + a_j (l+1)^\alpha + b_j (l+1)^\alpha \le r_j + a_j l^\alpha + a_k (l+1)^\alpha + b_k (l+1)^\alpha$ or $A + a_k l^\alpha + a_j (l+1)^\alpha + b_j (l+1)^\alpha \le r_j + a_j l^\alpha + b_j l^\alpha + b_k (l+1)^\alpha$ or $A + a_k l^\alpha + a_j (l+1)^\alpha + b_j (l+1)^\alpha \le r_k + a_k (l+1)^\alpha + b_k (l+1)^\alpha$

(viii) Either $a_k l^\alpha + b_k l^\alpha + b_j (l+1)^\alpha \le a_j l^\alpha + a_k (l+1)^\alpha + b_k (l+1)^\alpha$ or $a_k l^\alpha + b_k l^\alpha + b_j (l+1)^\alpha \le a_j l^\alpha + b_j l^\alpha + b_k (l+1)^\alpha$ or $A + a_k l^\alpha + b_k l^\alpha + b_j (l+1)^\alpha \le B + b_j l^\alpha + b_k (l+1)^\alpha$ or $A + a_k l^\alpha + b_k l^\alpha + b_j (l+1)^\alpha \le r_j + a_j l^\alpha + a_k (l+1)^\alpha + b_k (l+1)^\alpha$ or $A + a_k l^\alpha + b_k l^\alpha + b_j (l+1)^\alpha \le r_j + a_j l^\alpha + b_j l^\alpha + b_k (l+1)^\alpha$ or $A + a_k l^\alpha + b_k l^\alpha + b_j (l+1)^\alpha \le r_k + a_k (l+1)^\alpha + b_k (l+1)^\alpha$

(ix) Either $B + b_k l^\alpha + b_j (l+1)^\alpha \le A + a_j l^\alpha + a_k (l+1)^\alpha + b_k (l+1)^\alpha$ or $B + b_k l^\alpha + b_j (l+1)^\alpha \le A + a_j l^\alpha + b_j l^\alpha + b_k (l+1)^\alpha$ or $b_k l^\alpha + b_j (l+1)^\alpha \le b_j l^\alpha + b_k (l+1)^\alpha$ or $B + b_k l^\alpha + b_j (l+1)^\alpha \le r_j + a_j l^\alpha + a_k (l+1)^\alpha + b_k (l+1)^\alpha$ or $B + b_k l^\alpha + b_j (l+1)^\alpha \le r_j + a_j l^\alpha + b_j l^\alpha + b_k (l+1)^\alpha$ or $B + b_k l^\alpha + b_j (l+1)^\alpha \le r_k + a_k (l+1)^\alpha + b_k (l+1)^\alpha$

(x) Either $r_k + a_k l^\alpha + a_j (l+1)^\alpha + b_j (l+1)^\alpha \le A + a_j l^\alpha + a_k (l+1)^\alpha + b_k (l+1)^\alpha$ or $r_k + a_k l^\alpha + a_j (l+1)^\alpha + b_j (l+1)^\alpha \le A + a_j l^\alpha + b_j l^\alpha + b_k (l+1)^\alpha$ or $r_k + a_k l^\alpha + a_j (l+1)^\alpha + b_j (l+1)^\alpha \le B + b_j l^\alpha + b_k (l+1)^\alpha$ or $r_k + a_k l^\alpha + a_j (l+1)^\alpha + b_j (l+1)^\alpha \le r_j + a_j l^\alpha + a_k (l+1)^\alpha + b_k (l+1)^\alpha$ or $r_k + a_k l^\alpha + a_j (l+1)^\alpha + b_j (l+1)^\alpha \le r_j + a_j l^\alpha + b_j l^\alpha + b_k (l+1)^\alpha$ or $a_k l^\alpha + a_j (l+1)^\alpha + b_j (l+1)^\alpha \le a_k (l+1)^\alpha + b_k (l+1)^\alpha$

(xi) Either $r_k + a_k l^\alpha + b_k l^\alpha + b_j (l+1)^\alpha \le A + a_j l^\alpha + a_k (l+1)^\alpha + b_k (l+1)^\alpha$ or $r_k + a_k l^\alpha + b_k l^\alpha + b_j (l+1)^\alpha \le B + b_j l^\alpha + b_k (l+1)^\alpha$ or $r_k + a_k l^\alpha + b_k l^\alpha + b_j (l+1)^\alpha \le r_j + a_j l^\alpha + a_k (l+1)^\alpha + b_k (l+1)^\alpha$ or $r_k + a_k l^\alpha + b_k l^\alpha + b_j (l+1)^\alpha \le r_j + a_j l^\alpha + b_j l^\alpha + b_k (l+1)^\alpha$ or $a_k l^\alpha + b_k l^\alpha + b_j (l+1)^\alpha \le a_k (l+1)^\alpha + b_k (l+1)^\alpha$

(xii) Either $r_j + a_j (l+1)^\alpha + b_j (l+1)^\alpha \le A + a_j l^\alpha + a_k (l+1)^\alpha + b_k (l+1)^\alpha$ or $r_j + a_j (l+1)^\alpha + b_j (l+1)^\alpha \le A + a_j l^\alpha + b_j l^\alpha + b_k (l+1)^\alpha$ or $r_j + a_j (l+1)^\alpha + b_j (l+1)^\alpha \le B + b_j l^\alpha + b_k (l+1)^\alpha$ or $a_j (l+1)^\alpha + b_j (l+1)^\alpha \le a_j l^\alpha + a_k (l+1)^\alpha + b_k (l+1)^\alpha$ or $a_j (l+1)^\alpha + b_j (l+1)^\alpha \le a_j l^\alpha + b_j l^\alpha + b_k (l+1)^\alpha$ or $r_j + a_j (l+1)^\alpha + b_j (l+1)^\alpha \le r_k + a_k (l+1)^\alpha + b_k (l+1)^\alpha$

then S_2 dominates S_1.

Proof 1. Under schedules S_1 and S_2, the completion times of jobs J_j and J_k are

$$C_{1j}(S_1) = \max\{A, r_j\} + a_j l^\alpha, \tag{2}$$

$$\begin{aligned} C_j(S_1) &= \max\{B, C_{1j}(S_1)\} + b_j l^\alpha \\ &= \max\{r_j + a_j l^\alpha + b_j l^\alpha, A + a_j l^\alpha + b_j l^\alpha, B + b_j l^\alpha\}, \end{aligned} \tag{3}$$

$$\begin{aligned} C_{1k}(S_1) &= \max\{C_{1j}(S_1), r_k\} + a_k (l+1)^\alpha \\ &= \max\{r_j + a_j l^\alpha + a_k (l+1)^\alpha, \\ &\qquad A + a_j l^\alpha + a_k (l+1)^\alpha, r_k + a_k (l+1)^\alpha\}, \end{aligned} \tag{4}$$

$$\begin{aligned} C_k(S_1) &= \max\{C_{1k}(S_1), C_j(S_1)\} + b_k (l+1)^\alpha \\ &= \max\{A + a_j r^\alpha + a_k (l+1)^\alpha + b_k (l+1)^\alpha, \\ &\qquad A + a_j l^\alpha + b_j l^\alpha + b_k (l+1)\alpha, B + b_j l^\alpha \\ &\qquad + b_k (l+1)^\alpha, r_j + a_j l^\alpha + a_k (l+1)^\alpha \\ &\qquad + b_k (l+1)^\alpha, r_j + a_j l^\alpha + b_j l^\alpha + b_k (l+1)^\alpha, \\ &\qquad r_k + a_k (l+1)^\alpha + b_k (l+1)^\alpha\}, \end{aligned} \tag{5}$$

$$C_{1k}(S_2) = \max\{A, r_k\} + a_k l^\alpha, \tag{6}$$

$$C_k(S_2) = \max\{r_k + a_k l^\alpha + b_k l^\alpha, A + a_k l^\alpha + b_k l^\alpha, B + b_k l^\alpha\}, \tag{7}$$

$$\begin{aligned} C_{1j}(S_2) &= \max\{r_k + a_k l^\alpha + a_j (l+1)^\alpha, \\ &\qquad A + a_k l^\alpha + a_j (l+1)^\alpha, r_j + a_j (l+1)^\alpha\}, \end{aligned} \tag{8}$$

$$\begin{aligned} C_j(S_2) &= \max\{A + a_k l^\alpha + a_j (l+1)^\alpha + b_j (l+1)^\alpha, \\ &\qquad A + a_k l^\alpha + b_k l^\alpha + b_j (l+1)^\alpha, B + b_k l^\alpha \\ &\qquad + b_j (l+1)^\alpha, r_k + a_k l^\alpha + a_j (l+1)^\alpha \\ &\qquad + b_j (l+1)^\alpha, r_k + a_k l^\alpha + b_k l^\alpha + b_j (l+1)^\alpha, \\ &\qquad r_j + a_j (l+1)^\alpha + b_j (l+1)^\alpha\}. \end{aligned} \tag{9}$$

If the cases $r_k + a_k l^\alpha + b_k l^\alpha \le r_j + a_j l^\alpha + b_j l^\alpha$, $A + a_k l^\alpha + b_k l^\alpha \le r_j + a_j l^\alpha + b_j l^\alpha$, $B + b_k l^\alpha \le r_j + a_j l^\alpha + b_j l^\alpha$, $r_k + a_k l^\alpha + a_j (l+1)^\alpha \le r_j + a_j l^\alpha + a_k (l+1)^\alpha$, $r_k + a_k l^\alpha + a_j (l+1)^\alpha \le A +$

$a_j l^\alpha + a_k(l+1)^\alpha$, $A + a_k l^\alpha + a_j(l+1)^\alpha \le r_k + a_k(l+1)^\alpha$, $a_k l^\alpha + a_j(l+1)^\alpha + b_j(l+1)^\alpha \le a_j l^\alpha + a_k(l+1)^\alpha + b_k(l+1)^\alpha$, $a_k l^\alpha + b_k l^\alpha + b_j(l+1)^\alpha \le a_j l^\alpha + a_k(l+1)^\alpha + b_k(l+1)^\alpha$, $B + b_k l^\alpha + b_j(l+1)^\alpha \le A + a_j l^\alpha + a_k(l+1)^\alpha + b_k(l+1)^\alpha$, $r_k + a_k l^\alpha + a_j(l+1)^\alpha + b_j(l+1)^\alpha \le A + a_j l^\alpha + a_k(l+1)^\alpha + b_k(l+1)^\alpha$, $r_k + a_k l^\alpha + b_k l^\alpha + b_j(l+1)^\alpha \le A + a_j l^\alpha + a_k(l+1)^\alpha + b_k(l+1)^\alpha$, and $r_j + a_j(l+1)^\alpha + b_j(l+1)^\alpha \le A + a_j l^\alpha + a_k(l+1)^\alpha + b_k(l+1)^\alpha$ can be satisfied, we have the following: first term in (7) ≤ first term in (3), second term in (7) ≤ first term in (3), third term in (7) ≤ first term in (3), first term in (8) ≤ first term in (4), second term in (8) ≤ first term in (4), third term in (8) ≤ first term in (4), first term in (9) ≤ first term in (5), second term in (9) ≤ first term in (5), third term in (9) ≤ first term in (5), fourth term in (9) ≤ first term in (5), fifth term in (9) ≤ first term in (5), and sixth term in (9) ≤ first term in (5).

Hence, $C_{1j}(S_2) \le C_{1k}(S_1)$, $C_k(S_2) \le C_j(S_1)$, $C_j(S_2) \le C_k(S_1)$, and $C_k(S_2) + C_j(S_2) \le C_j(S_1) + C_k(S_1)$. Similarly to this case, the other cases can be obtained.

4. Branch-and-Bound (B&B) Algorithm

4.1. Lower Bound. Let $\pi = (\pi_1, \pi_2)$ be a schedule in which π_1 is the scheduled part, and suppose there are θ jobs in π_1, and π_2 is the set of so far unscheduled jobs. Let $p_{i(j)}(\theta+1 \le j \le n)$ denote the jth smallest normal processing time on machine M_i ($i = 1$ and 2) when jobs in π_2 are arranged in an ascending order of the normal processing time on machine M_i. By definition, the completion time of the $(\theta+1)$th job on machine M_2 is

$$\begin{aligned} C_{2[\theta+1]}(\pi) &= \max\left\{\max\left\{r_{[\theta+1]}, C_{1[\theta]}(\pi)\right\} + a_{[\theta+1]}(\theta+1)^\alpha, C_{2[\theta]}(\pi)\right\} + b_{[\theta+1]}(\theta+1)^\alpha \\ &= \max\left\{r_{[\theta+1]} + a_{[\theta+1]}(\theta+1)^\alpha + b_{[\theta+1]}(\theta+1)^\alpha, C_{1[\theta]}(\pi) + a_{[\theta+1]}(\theta+1)^\alpha + b_{[\theta+1]}(\theta+1)^\alpha, C_{2[\theta]}(\pi) + b_{[\theta+1]}(\theta+1)^\alpha\right\} \\ &\ge r_{[\theta+1]}\left(a_{[\theta+1]} + b_{[\theta+1]}\right)(\theta+1)^\alpha. \end{aligned} \tag{10}$$

Similarly,

$$C_{2[\theta+j]}(\pi) \ge r_{[\theta+j]} + \left(a_{[\theta+j]} + b_{[\theta+j]}\right)(\theta+j)^\alpha, \tag{11}$$

where $1 \le j \le n-\theta$.

Hence,

$$C_{\max} = C_{2[n]}(\pi) \ge r_{\max} + \overline{ab}_{\min} n^\alpha, \tag{12}$$

where $r_{\max} = \max\{r_j|_{j\in\pi_2}\}$ and $\overline{ab}_{\min} = \min\{a_j + b_j|_{j\in\pi_2}\}$.

$$\begin{aligned} \sum_{j=1}^{n} C_j(\pi) &= \sum_{j=1}^{\theta} C_{2[j]}(\pi) + \sum_{j=\theta+1}^{n} C_{2[j]}(\pi) \\ &\ge \sum_{j=1}^{\theta} C_{2[j]}(\pi) + \sum_{j=1}^{n-\theta} r_{[\theta+j]} + \sum_{j=1}^{n-\theta}\left(a_{[\theta+j]} + b_{[\theta+j]}\right)(\theta+j)^\alpha. \end{aligned} \tag{13}$$

Hence,

$$\begin{aligned} &\lambda C_{\max} + (1-\lambda)\sum_{j=1}^{n} C_j(\pi) \\ &\ge \lambda\left(r_{\max} + \overline{ab}_{\min} n^\alpha\right) + (1-\lambda) \\ &\quad \times \left(\sum_{j=1}^{\theta} C_{2[j]}(\pi) + \sum_{j=1}^{n-\theta} r_{[\theta+j]} + \sum_{j=1}^{n-\theta}\left(a_{[\theta+j]} + b_{[\theta+j]}\right)(\theta+j)^\alpha\right) \\ &= \lambda\left(r_{\max} + \overline{ab}_{\min} n^\alpha\right) + (1-\lambda)\left(\sum_{j=1}^{\theta} C_{2[j]}(\pi) + \sum_{j=1}^{n-\theta} r_{[\theta+j]}\right) \\ &\quad + (1-\lambda)\left(\sum_{j=1}^{n-\theta}\left(a_{[\theta+j]} + b_{[\theta+j]}\right)(\theta+j)^\alpha\right). \end{aligned} \tag{14}$$

It is noticed that the term $\lambda(r_{\max} + \overline{ab}_{\min} n^\alpha) + (1-\lambda)(\sum_{j=1}^{\theta} C_{2[j]}(\pi) + \sum_{j=1}^{n-\theta} r_{[\theta+j]})$ is a fixed constant and the term $(1-\lambda)(\sum_{j=1}^{n-\theta}(a_{[\theta+j]} + b_{[\theta+j]})(\theta+j)^\alpha)$ is minimized by sequencing the jobs in a nondecreasing order of $a_{[j]} + b_{[j]}$. Consequently, we obtain the first lower bound which is

$$\begin{aligned} \mathrm{LB}_1 &= \lambda\left(r_{\max} + \overline{ab}_{\min} n^\alpha\right) + (1-\lambda)\left(\sum_{j=1}^{\theta} C_{2[j]}(\pi) + \sum_{j=1}^{n-\theta} r_{[\theta+j]}\right) \\ &\quad + (1-\lambda)\left(\sum_{j=1}^{n-\theta}\left(a_{(\theta+j)} + b_{(\theta+j)}\right)(\theta+j)^\alpha\right), \end{aligned} \tag{15}$$

where $r_{\max} = \max\{r_j \mid j \in \pi_2\}$, $\overline{ab}_{\min} = \min\{a_j + b_j \mid j \in \pi_2\}$, and $a_{(\theta+1)} + b_{(\theta+1)} \le a_{(\theta+2)} + b_{(\theta+2)} \le \cdots \le a_{(n)} + b_{(n)}$.

Similarly,

$$C_{2[\theta+j]}(\pi) \ge C_{1[\theta]}(\pi) + \sum_{l=1}^{j} a_{[\theta+l]}(\theta+l)^\alpha + b_{[\theta+j]}(\theta+j)^\alpha, \tag{16}$$

where $1 \le j \le n-\theta$.

Hence,

$$C_{\max} \ge C_{1[\theta]}(\pi) + \sum_{l=1}^{n-\theta} a_{[\theta+l]}(\theta+l)^\alpha + b_{[n]} n^\alpha, \tag{17}$$

and the total completion time is

$$\begin{aligned}\sum_{j=1}^{n} C_j(\pi) &= \sum_{j=1}^{\theta} C_{2[j]}(\pi) + \sum_{j=\theta+1}^{n} C_{2[j]}(\pi) \\ &\geq \sum_{j=1}^{\theta} C_{2[j]}(\pi) + (n-\theta)C_{1[\theta]}(\pi) \\ &\quad + \sum_{j=1}^{n-\theta}\sum_{l=1}^{j} a_{[\theta+l]}(\theta+l)^{\alpha} + \sum_{j=1}^{n-\theta} b_{[\theta+j]}(\theta+j)^{\alpha}.\end{aligned} \tag{18}$$

Hence,

$$\begin{aligned}&\lambda C_{\max} + (1-\lambda)\sum_{j=1}^{n} C_j(\pi) \\ &\geq (1-\lambda)\sum_{j=1}^{\theta} C_{2[j]}(\pi) + (\lambda + (1-\lambda)(n-\theta))C_{1[\theta]}(\pi) \\ &\quad + \lambda\left(\sum_{l=1}^{n-\theta} a_{[\theta+l]}(\theta+l)^{\alpha} + b_{[n]}n^{\alpha}\right) \\ &\quad + (1-\lambda)\left(\sum_{j=1}^{n-\theta}\sum_{l=1}^{j} a_{[\theta+l]}(\theta+l)^{\alpha} + \sum_{j=1}^{n-\theta} b_{[\theta+j]}(\theta+j)^{\alpha}\right).\end{aligned} \tag{19}$$

Obviously, $(1-\lambda)\sum_{j=1}^{\theta} C_{2[j]}(\pi) + (\lambda + (1-\lambda)(n-\theta))C_{1[\theta]}(\pi)$ is a fixed constant, $b_{[n]}n^{\alpha}$ is minimized by choosing $b_{\min} = \min\{b_j|_{j\in\pi_2}\}$ and $\lambda(\sum_{l=1}^{n-\theta} a_{[\theta+l]}(\theta+l)^{\alpha})$, and $(1-\lambda)(\sum_{j=1}^{n-\theta}\sum_{l=1}^{j} a_{[\theta+l]}(\theta+l)^{\alpha} + \sum_{j=1}^{n-\theta} b_{[\theta+j]}(\theta+j)^{\alpha})$ can be minimized by sequencing the jobs in a nondecreasing order of $a_{[j]}$ and $b_{[j]}$, respectively. Consequently, we have

$$\begin{aligned}\mathrm{LB}_2 &= (1-\lambda)\sum_{j=1}^{\theta} C_{2[j]}(\pi) + (\lambda + (1-\lambda)(n-\theta))C_{1[\theta]}(\pi) \\ &\quad + \lambda\left(\sum_{l=1}^{n-\theta} a_{(\theta+l)}(\theta+l)^{\alpha} + b_{\min}n^{\alpha}\right) \\ &\quad + (1-\lambda)\left(\sum_{j=1}^{n-\theta}\sum_{l=1}^{j} a_{(\theta+l)}(\theta+l)^{\alpha} + \sum_{j=1}^{n-\theta} b_{(\theta+j)}(\theta+j)^{\alpha}\right),\end{aligned} \tag{20}$$

where $b_{\min} = \min\{b_j \mid j\in\pi_2\}$ and $a_{(\theta+1)} \leq a_{(\theta+2)} \leq \cdots \leq a_{(n)}$ $(b_{(\theta+1)} \leq b_{(\theta+2)} \leq \cdots \leq b_{(n)})$ is a nondecreasing order of the normal processing times on $M_1(M_2)$ for the remaining unscheduled jobs (note that $a_{(l)}$ and $b_{(l)}$ do not necessarily correspond to the same job).

Similarly,

$$C_{2[\theta+j]}(\pi) \geq C_{2[\theta]}(\pi) + \sum_{l=1}^{j} b_{[\theta+l]}(\theta+l)^{\alpha}, \tag{21}$$

where $1 \leq j \leq n-\theta$.

Hence,

$$C_{\max} \geq C_{2[\theta]}(\pi) + \sum_{l=1}^{n-\theta} b_{[\theta+l]}(\theta+l)^{\alpha}, \tag{22}$$

and

$$\begin{aligned}\sum_{j=1}^{n} C_j(\pi) &= \sum_{j=1}^{\theta} C_{2[j]}(\pi) + \sum_{j=\theta+1}^{n} C_{2[j]}(\pi) \\ &\geq \sum_{j=1}^{\theta} C_{2[j]}(\pi) + (n-\theta)C_{2[\theta]}(\pi) \\ &\quad + \sum_{j=1}^{n-\theta}\sum_{l=1}^{j} b_{[\theta+j]}(\theta+l)^{\alpha}.\end{aligned} \tag{23}$$

Hence,

$$\begin{aligned}&\lambda C_{\max} + (1-\lambda)\sum_{j=1}^{n} C_j(\pi) \\ &\geq (1-\lambda)\sum_{j=1}^{\theta} C_{2[j]}(\pi) + (\lambda + (1-\lambda)(n-\theta))C_{2[\theta]}(\pi) \\ &\quad + \lambda\left(\sum_{l=1}^{n-\theta} b_{[\theta+l]}(\theta+l)^{\alpha}\right) + (1-\lambda)\left(\sum_{j=1}^{n-\theta}\sum_{l=1}^{j} b_{[\theta+l]}(\theta+l)^{\alpha}\right).\end{aligned} \tag{24}$$

It is noticed that the term $(1-\lambda)\sum_{j=1}^{\theta} C_{2[j]}(\pi) + (\lambda + (1-\lambda)(n-\theta))C_{2[\theta]}(\pi)$ is a fixed constant and the term $\lambda(\sum_{l=1}^{n-\theta} b_{[\theta+l]}(\theta+l)^{\alpha}) + (1-\lambda)(\sum_{j=1}^{n-\theta}\sum_{l=1}^{j} b_{[\theta+l]}(\theta+l)^{\alpha})$ can be minimized by sequencing the jobs in a nondecreasing order of $b_{[j]}$. Consequently, we have

$$\begin{aligned}LB_3 &= (1-\lambda)\sum_{j=1}^{\theta} C_{2[j]}(\pi) + (\lambda + (1-\lambda)(n-\theta))C_{2[\theta]}(\pi) \\ &\quad + \lambda\left(\sum_{l=1}^{n-\theta} b_{(\theta+l)}(\theta+l)^{\alpha}\right) \\ &\quad + (1-\lambda)\left(\sum_{j=1}^{n-\theta}\sum_{l=1}^{j} b_{(\theta+l)}(\theta+l)^{\alpha}\right),\end{aligned} \tag{25}$$

where $b_{(\theta+1)} \leq b_{(\theta+2)} \leq \cdots \leq b_{(n)}$ is a nondecreasing order of the normal processing times on M_2 for the remaining unscheduled jobs.

Phase I
Step 1. Sequence the jobs in nondecreasing order of r_j
Step 2. Sequence the jobs in nondecreasing order of a_j
Step 3. Sequence the jobs in nondecreasing order of b_j.
Step 4. Sequence the jobs in nondecreasing order of $a_j + b_j$
Step 5. Choose the best solution from Steps 1 to 4
Phase II
Step 1. Let π_0 be the schedule obtained from Phase I
Step 2. Set $k = 2$. Select the first two jobs from π_0 and select the better between the two possible sequences
Step 3. Increment k, $k = k + 1$. Select the kth job from π_0 and insert it into k possible positions of the best partial sequence obtained so far. Among the k sequences, the best k-job partial sequence is selected based on minimum $\lambda C_{\max} + (1 - \lambda) \sum_{j=1}^{n} C_j$. Next, determine all possible sequences by interchanging jobs in positions i and j of the above partial sequence for all i, $j(1 \le i < k, i < j \le k)$. Select the best partial sequence among $k(k-1)/2$ sequences having minimum $\lambda C_{\max} + (1 - \lambda) \sum_{j=1}^{n} C_j$
Step 4. If $k = n$, then STOP; else, go to Step 3

Algorithm 1: FL-based two-phase algorithm (FLTPA).

Table 1: Results of the B&B algorithm and the FLTPA heuristic for r_j~[1, 10].

n	α	CPU time (ms)				Node number		Error	
		FLTPA		B&B					
		Mean	Max	Mean	Max	Mean	Max	Mean (%)	Max (%)
	−0.152	0	0	44.320	172	1265.940	5689	0.131	1.792
11	−0.322	0.94	16	45.260	280	1283.260	8116	0.120	1.777
	−0.515	0.64	16	16.520	187	445.660	5037	0.058	1.151
	−0.152	0.62	16	151.340	1404	4556.020	44563	0.129	2.105
12	−0.322	0.3	15	70.540	359	1930.160	9557	0.211	2.973
	−0.515	0.62	16	52.420	390	1401.980	9733	0.162	2.043
	−0.152	0.96	16	385.920	2699	9631.740	76047	0.220	1.798
13	−0.322	1.26	16	119.180	780	2858.480	17116	0.255	2.622
	−0.515	0.64	16	119.800	655	2883.180	15506	0.177	2.497
	−0.152	2.16	16	1518.840	19,890	36283.359	466540	0.228	2.435
14	−0.322	1.56	16	543.500	4197	12781.740	96673	0.240	1.913
	−0.515	2.18	16	202.180	1872	4837.700	47663	0.147	1.650
	−0.152	3.12	16	4629.780	118,311	102398.320	2724303	0.198	2.759
15	−0.322	1.82	16	1299.860	13,869	30514.420	317594	0.157	0.970
	−0.515	1.54	16	424.020	5382	10577.960	141059	0.114	0.622

In order to make the lower bound tighter, we choose the maximum values LB_h, $h = 1,2,3$, as a lower bound for π_2, i.e.,

$$LB = \max \{LB_1, LB_2, LB_3\}. \quad (26)$$

4.2. The Algorithm for the Upper Bound. Similarly to Framinan and Leisten [40] (they proposed the $O(mn^4)$ FL heuristic for solving $Fm|prmu|\sum C_j$), the FL heuristic can be adjusted for solving $F2|\text{LE}, r_j|\lambda C_{\max} + (1 - \lambda)\sum_{j=1}^{n} C_j$. Now, we give the modified FL algorithm.

Since Phase I takes $O(n \log n)$ time and Phase II takes $O(mn^4)$ time, the overall time complexity of the FLTPA algorithm is $O(mn^4)$.

4.3. The B&B Algorithm. A B&B algorithm that uses the depth first search strategy to solve $F2|\text{LE}, r_j|\lambda C_{\max} + (1 - \lambda) \sum_{j=1}^{n} C_j$ is proposed. Now the B&B algorithm can be described as follows.

Step 1. Initialization: the FL-based two-phase algorithm (FLTPA) is applied to obtain an upper bound.

Step 2. Fathoming: apply Proposition 1 to eliminate the dominated partial sequences from the initial node and their descendants from the tree.

Step 3. Bounding: calculate the lower bound for the node. If the lower bound for an unfathomed partial schedule is larger

Table 2: Results of the B&B algorithm and the FLTPA heuristic for r_j~[1, 50].

n	α	CPU time (ms)				Node number		Error	
		FLTPA		B&B					
		Mean	Max	Mean	Max	Mean	Max	Mean	Max
11	−0.152	1.56	16	38.380	219	1120.480	7152	0.301	4.920
	−0.322	0	0	22.480	156	623.360	6093	0.117	1.076
	−0.515	0.3	15	15.300	47	402.100	1456	0.314	4.642
12	−0.152	0.92	16	110.780	1341	3182.240	40970	0.195	2.423
	−0.322	0.94	16	66.140	203	1872.820	6067	0.263	3.295
	−0.515	1.26	16	40.860	656	1191.280	21210	0.301	2.267
13	−0.152	1.86	16	274.580	2012	6466.040	43732	0.355	3.760
	−0.322	1.6	16	216.180	1186	5390.040	32324	0.375	4.117
	−0.515	1.26	16	142.260	1107	3474.940	25696	0.538	5.303
14	−0.152	1.28	16	743.780	5725	17542.561	124227	0.488	2.817
	−0.322	3.76	16	272.040	1981	6133.940	43542	0.112	0.942
	−0.515	1.58	16	477.660	8361	11843.340	213835	0.480	3.439
15	−0.152	2.84	16	1422.080	10,983	32341.820	272559	0.256	2.603
	−0.322	1.88	16	1106.980	12,043	26354.939	285013	0.272	1.663
	−0.515	1.84	16	461.180	3854	11786.160	118523	0.207	2.318

Table 3: Results of the B&B algorithm and the FLTPA heuristic for r_j~[1,100].

n	α	CPU time (ms)				Node number		Error	
		FLTPA		B&B					
		Mean	Max	Mean	Max	Mean	Max	Mean	Max
11	−0.152	0.62	16	50.860	343	1288.380	11,443	0.634	4.170
	−0.322	1.56	16	27.460	94	735.060	2719	0.528	5.771
	−0.515	0	0	18.100	125	436.120	3457	0.481	3.671
12	−0.152	1.58	16	108.580	998	2754.880	25142	0.377	3.067
	−0.322	0.62	16	46.180	297	1122.220	8368	0.321	1.720
	−0.515	1.24	16	50.560	546	1296.280	15767	0.602	4.513
13	−0.152	1.24	16	167.860	920	4312.520	24715	0.365	2.844
	−0.322	1.9	16	132.260	1809	3278.320	45866	0.560	3.596
	−0.515	1.28	16	94.200	1061	2279.440	26942	0.507	4.397
14	−0.152	1.56	16	433.060	3775	10215.620	97348	0.505	3.298
	−0.322	3.46	16	196.220	1170	4658.940	29477	0.550	4.448
	−0.515	1.58	16	191.860	1279	4328.420	26003	0.528	4.147
15	−0.152	2.52	16	1491.960	34,819	33631.441	775620	0.586	3.553
	−0.322	2.14	16	774.120	8954	17432.561	224454	0.400	2.774
	−0.515	3.46	16	561.260	9579	12746.040	211568	0.607	5.383

than the upper bound, eliminate the node and all the nodes following it in the branch. Calculate the objective function value $(\lambda C_{\max} + (1-\lambda)\sum_{j=1}^{n} C_j)$ of the completed schedule; if it is less than the upper bound, replace it as the new solution; otherwise, eliminate it.

Step 4. Termination: continue to search all the nodes, and the remaining upper bound is optimal.

4.4. The Improved EMO Algorithm. The B&B algorithm outputs the optimal solution, and the LFTPA heuristic outputs a near-optimal solution to the linearly weighted sum of $C_{\max}$ and $\sum_{j=1}^{n} C_j$. From the perspective of biobjective optimization, evolutionary multiobjective optimization (EMO) algorithm returns at one time a set of nondominated solutions [41–43], i.e., the Pareto front, for the decision-maker's reference. We propose two strategies to improve the multiobjective memetic algorithm (MOMA) which has successfully solved many difficult numerical optimization problems and outperforms NSGA-II (the nondominated sorting genetic algorithm) and SPEA2 (the improved strength Pareto evolutionary algorithm) for the

two-objective and three-objective benchmark flow shop scheduling instances [44].

(1) Initialization strategy: the initial population consists of one individual generated by the LFTPA heuristic and some others adjusted from that individual by exchanging operators. The number of the LFTPA heuristic-related individuals is equal to half of the population size. The remaining individuals are randomly generated to guarantee population diversity.

(2) Global search strategy: due to the nondominated sorting and crowd distance calculation mechanisms, NSGA-II has a very good performance to solve many multiobjective problems and is used as global search. Its key operators include selection, crossover, and mutation. We apply the tournament selection operator (see the details in [44]). According to Ishibuchi et al. [45], the two-point crossover and insertion mutation operators are adopted.

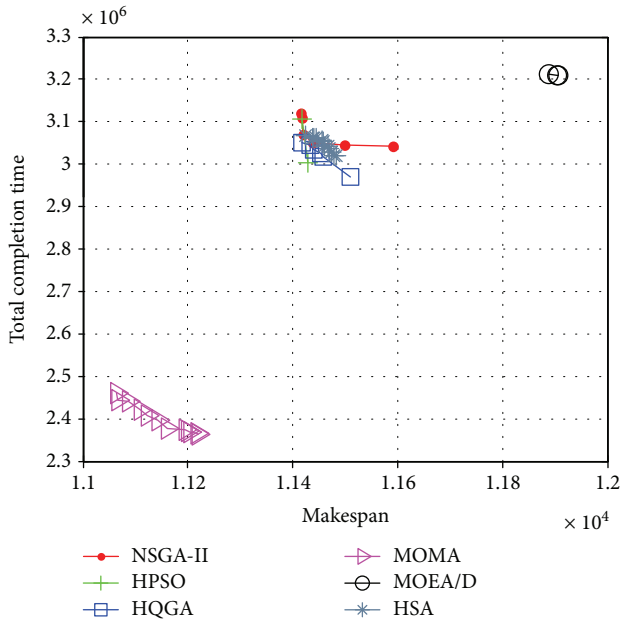

Figure 1: Pareto front of a problem instance for $n = 500$, $r_j \sim [1, 50]$, and $\alpha = -0.152$.

5. Computational Study

Computational experiments were studied to evaluate the effectiveness of the B&B algorithm and the FLTPA heuristic. We coded the B&B algorithm, the FLTPA heuristic, and the improved MOMA in VC++ 6.0 and ran on CPU Intel® Core (TM) i7-4790 3.6 GHz and 8 GB RAM. For the experiments, the parameters are considered as follows: $\lambda = 0.5$ and the numbers of jobs (n): 11, 12, 13, 14, and 15 for small size and 500 and 1000 for large size. The normal processing times of jobs (a_j and b_j) are randomly generated from the uniform distribution over the integers [1, 100].

In order to study the impact of the parameters, the values of the learning effect are taken to be 90% (i.e., $\alpha = \log_2 0.9 = -0.152$), 80% (i.e., $\alpha = \log_2 0.8 = -0.322$), and 70% (i.e., $\alpha = \log_2 0.7 = -0.515$). The release dates of jobs (r_j) are randomly generated from the uniform distribution over the integers [1, 10,1, 50] and [1, 100].

As a consequence, for small-size problems, 45 experimental conditions were examined and 50 replications were randomly generated for each condition. A total of 2250 problems were tested. For large-size problems, 18 experimental conditions were examined and 50 replications were randomly generated for each condition. A total of 900 problems were tested. For the B&B algorithm, the average and maximum number of nodes and the average and the maximum time (in milliseconds) are reported. The percentage error of the solution produced by the FLTPA heuristic is calculated as

$$\frac{\lambda C_{\max}(\text{FLTPA}) + (1-\lambda)\sum_{j=1}^{n} C_j(\text{FLTPA}) - \left(\lambda C_{\max}(\text{OPT}) + (1-\lambda)\sum_{j=1}^{n} C_j(\text{OPT})\right)}{\lambda C_{\max}(\text{OPT}) + (1-\lambda)\sum_{j=1}^{n} C_j(\text{OPT})} \times 100\%. \quad (27)$$

The results of small-size problems are summarized in Tables 1–3. From Tables 1–3, we find that the B&B algorithm can solve a problem of up to 15 jobs in a very short amount of time. The most time-consuming instance took a maximum of 118311 ms for $n = 15$, $\alpha = -0.152$, and $r_j \sim [1, 10]$. As for the performance of the FLTPA heuristic, it is seen that the FLTPA heuristic performs very well for the error percentages. The mean error percentage is less than 0.7% for all the tested cases, and the max error percentage is less than 6% for all the tested cases.

For large-size problems, the termination of the proposed MOMA algorithm is controlled by setting a running time limit of 100 seconds. In order to verify the effectiveness and efficiency, the MOMA is compared with several state-of-the-art EMO algorithms [44], including the hybrid quantum-inspired genetic algorithm (HQGA), the hybrid particle swarm optimization algorithm (HPSO), the hybrid simulated annealing algorithm (HSA), the multiobjective evolutionary algorithm based on decomposition (MOEA/D), and NSGA-II. The parameters of the above algorithms are taken from Liu et al. [44]. The performance of an EMO is reflected by the obtained Pareto front's proximity and diversity. We consider two metrics for performance measurement. The IGD (inverted generational distance)

Table 4: The mean IGD value ($\times 10^6$) of 6 algorithms.

n	r_j	α	NSGA-II	HPSO	HQGA	HSA	MOEA/D	MOMA
		−0.152	6.653	6.770	6.518	6.597	6.569	**0.000**
	[1, 10]	−0.322	3.971	4.091	3.866	3.931	4.140	**0.000**
		−0.515	2.197	2.308	2.136	2.179	1.961	**0.000**
		−0.152	6.713	6.962	6.683	6.755	7.046	**0.027**
500	[1, 50]	−0.322	3.838	3.995	3.767	3.817	4.248	**0.000**
		−0.515	2.133	2.228	2.098	2.137	2.113	**0.000**
		−0.152	6.635	6.827	6.506	6.630	6.946	**0.004**
	[1, 100]	−0.322	3.837	4.002	3.771	3.796	4.095	**0.000**
		−0.515	2.065	2.142	2.034	2.062	2.128	**0.000**
		−0.152	2.781	2.799	2.738	2.780	2.223	**0.001**
	[1, 10]	−0.322	1.455	1.478	1.450	1.448	0.815	**0.000**
		−0.515	7.305	7.422	7.264	7.279	5.803	**0.000**
		−0.152	2.707	2.708	2.655	2.689	1.993	**0.001**
1000	[1, 50]	−0.322	1.431	1.454	1.413	1.423	0.968	**0.001**
		−0.515	7.132	7.285	7.071	7.131	5.075	**0.000**
		−0.152	2.738	2.739	2.688	2.736	1.885	**0.000**
	[1, 100]	−0.322	1.428	1.458	1.406	1.422	0.860	**0.000**
		−0.515	6.917	7.035	6.826	6.927	5.706	**0.000**

Table 5: The mean HV values ($\times 10^{10}$) of 6 algorithms.

n	r_j	α	NSGA-II	HPSO	HQGA	HSA	MOEA/D	MOMA
		−0.152	0.619	0.593	0.646	0.632	0.586	**2.123**
	[1, 10]	−0.322	1.300	1.187	1.404	1.343	0.985	**6.742**
		−0.515	0.332	0.278	0.365	0.342	0.427	**2.195**
		−0.152	0.693	0.644	0.697	0.686	0.560	**2.187**
500	[1, 50]	−0.322	1.562	1.416	1.636	1.586	0.997	**7.057**
		−0.515	0.359	0.311	0.380	0.361	0.306	**2.186**
		−0.152	0.633	0.599	0.663	0.638	0.543	**2.091**
	[1, 100]	−0.322	0.160	0.143	0.166	0.163	0.115	**0.714**
		−0.515	0.390	0.353	0.415	0.392	0.274	**2.172**
		−0.152	0.298	0.292	0.310	0.298	0.480	**1.318**
	[1, 10]	−0.322	0.504	0.475	0.513	0.510	1.721	**3.663**
		−0.515	0.095	0.087	0.097	0.097	0.222	**0.970**
		−0.152	0.300	0.298	0.314	0.303	0.535	**1.320**
1000	[1, 50]	−0.322	0.516	0.490	0.542	0.530	1.396	**3.648**
		−0.515	0.099	0.089	0.102	0.098	0.286	**0.944**
		−0.152	0.301	0.299	0.315	0.302	0.587	**1.308**
	[1, 100]	−0.322	0.507	0.467	0.536	0.514	1.526	**3.581**
		−0.515	0.102	0.095	0.108	0.101	0.196	**0.915**

mainly evaluates the proximity of a Pareto front to the optimal reference set, which consists of nondominated solutions in the grouped Pareto fronts. The smaller the IGD metric is, the better proximity a Pareto front has. And the HV (hypervolume) evaluates the maximal area dominated by a Pareto front, which estimates both the convergence and diversity. The larger the value of HV means the better the integrated performance. As is shown in Figure 1 (Pareto front of a problem instance for $n = 500$, $r_j \sim [1, 50]$, and $\alpha = -0.152$), the proposed MOMA clearly dominates the rest of the algorithms. This result is similar for the rest of the problem instances. The dominance relationship is further demonstrated in

Table 6: Tests of between-subject effects to the FLTPA heuristic's error.

Source	Type III sum of squares	df	Mean square	F	Sig.
Intercept	0.024	1	0.024	434.996	0.000
Error	0.003	49	5.50×10^{-5}		
r_j	0.004	1.738	0.002	64.339	**0.000**
n	0.000	4	7.15×10^{-5}	1.949	0.104
α	9.51×10^{-5}	2	4.75×10^{-5}	1.035	0.359
$r_j * n$	0.000	6.094	6.14×10^{-5}	1.083	0.373
$r_j * \alpha$	0.000	3.191	8.99×10^{-5}	1.632	0.181
$n * \alpha$	0.000	8	3.95×10^{-5}	0.777	0.623
$r_j * n * \alpha$	0.001	10.777	5.91×10^{-5}	0.926	0.513

Table 7: Tests of between-subject effects to IGD of MOMA.

Source	Type III sum of squares	df	Mean square	F	Sig.
Intercept	84615121.839	1.000	84615121.839	5.971	0.018
Error	694370982.111	49.000	14170836.370		
n	1080878.541	1.000	1080878.541	0.071	0.791
r_j	45480148.228	1.445	31473956.059	1.525	0.226
α	130844023.359	1.060	123491834.545	4.492	0.037
$r_j * n$	29827527.174	1.440	20717814.407	0.967	0.360
$r_j * \alpha$	10336869.257	1.058	9768265.055	0.342	0.573
$n * \alpha$	54777091.926	1.536	35666809.099	0.907	0.385
$r_j * n * \alpha$	97890760.041	1.544	63387257.520	1.607	0.211

Table 8: Tests of between-subject effects to HV of MOMA.

Source	Type III sum of squares	df	Mean square	F	Sig.
Intercept	1.07913×10^{22}	1.000	1.07913×10^{22}	96208.156	**0.000**
Error	5.49614×10^{18}	49.000	1.12166×10^{17}		
n	5.3842×10^{21}	1.000	5.3842×10^{21}	34294.873	**0.000**
r_j	4.11552×10^{17}	1.960	2.09991×10^{17}	1.118	0.330
α	8.24917×10^{21}	1.214	6.79569×10^{21}	33897.141	**0.000**
$r_j * n$	2.46247×10^{17}	2.000	1.23123×10^{17}	0.806	0.450
$r_j * \alpha$	4.41039×10^{21}	1.137	3.87975×10^{21}	16984.584	**0.000**
$n * \alpha$	2.48526×10^{17}	2.323	1.06983×10^{17}	0.366	0.726
$r_j * n * \alpha$	3.88625×10^{16}	2.376	1.63594×10^{16}	0.067	0.957

Tables 4 and 5. MOMA's smallest IGD values and largest HV values of all instances show that it comprehensively outperforms the other 5 algorithms.

Finally, we examine the impact of choice of parameters, n, r_j, and α, on algorithm performances. We perform a N-way ANOVA using the commercial software SPSS version 17.0. The impacts of n, r_j, and α on the FLTPA heuristic's error, MOMA's IGD metric, and MOMA's HV metric are shown in Tables 6–8, respectively. Throughout the results, when the significance level (the last column in the tables) is under 0.01, the impact is deemed as significant. We observe that r_j is significant for the FLTPA heuristic's performance. The error percentage is much smaller when jobs' release dates are smaller. This could be explained by smaller release dates that do not tightly restrict the solution space. We further find out that all parameters are not significant for MOMA's IGD metric, whereas n, α, and $n * \alpha$ all have a significant impact on MOMA's HV metric. As n and/or α change, the hypervolume of a Pareto front would change dramatically as large numerical values are involved.

6. Summary and Future Research

In this study, we considered a bicriterion makespan and total completion time minimization flow shop scheduling with a learning effect subject to release dates. We propose a branch-and-bound algorithm and a heuristic algorithm. Numerical studies showed that the FLTPA heuristic was shown to perform well in obtaining near-optimal solutions. Future research may focus on considering the other bicriteria flow shop scheduling problems with a learning effect subject to release dates or studying more efficient heuristic algorithms.

Data Availability

Anyone who is interested in obtaining the data underlying the findings can send emails to Professor Jian Xu for help.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was supported by the National Natural Science Foundation of China (no. 71471120, 71872034) and the Support Program for Innovative Talents in Liaoning University (LR2016017). The Liaoning BaiQianWan Talents Program also sponsored this research.

References

[1] D. Biskup, "Single-machine scheduling with learning considerations," *European Journal of Operational Research*, vol. 115, no. 1, pp. 173–178, 1999.

[2] D. Biskup, "A state-of-the-art review on scheduling with learning effects," *European Journal of Operational Research*, vol. 188, no. 2, pp. 315–329, 2008.

[3] Y. Yin, K. E. Stecke, M. Swink, and I. Kaku, "Lessons from *seru* production on manufacturing competitively in a high cost environment," *Journal of Operations Management*, vol. 49-51, pp. 67–76, 2017.

[4] F. Al-Anzi and A. Allahverdi, "The relationship between three-tiered client-server internet database connectivity and two-machine flowshop," *International Journal of Parallel and Distributed Systems and Networks*, vol. 4, pp. 94–101, 2001.

[5] A. Allahverdi and F. S. Al-Anzi, "Using two-machine flowshop with maximum lateness objective to model multimedia data objects scheduling problem for WWW applications," *Computers & Operations Research*, vol. 29, no. 8, pp. 971–994, 2002.

[6] V. Fernandez-Viagas and J. M. Framinan, "NEH-based heuristics for the permutation flowshop scheduling problem to minimise total tardiness," *Computers & Operations Research*, vol. 60, pp. 27–36, 2015.

[7] J. M. Framinan, J. N. D. Gupta, and R. Leisten, "A review and classification of heuristics for permutation flow-shop scheduling with makespan objective," *Journal of the Operational Research Society*, vol. 55, no. 12, pp. 1243–1255, 2004.

[8] Q.-K. Pan and R. Ruiz, "A comprehensive review and evaluation of permutation flowshop heuristics to minimize flowtime," *Computers & Operations Research*, vol. 40, no. 1, pp. 117–128, 2013.

[9] R. Ruiz and C. Maroto, "A comprehensive review and evaluation of permutation flowshop heuristics," *European Journal of Operational Research*, vol. 165, no. 2, pp. 479–494, 2005.

[10] E. Vallada, R. Ruiz, and G. Minella, "Minimising total tardiness in the *m*-machine flowshop problem: a review and evaluation of heuristics and metaheuristics," *Computers & Operations Research*, vol. 35, no. 4, pp. 1350–1373, 2008.

[11] Z. Xu, L. Sun, and J. Gong, "Worst-case analysis for flow shop scheduling with a learning effect," *International Journal of Production Economics*, vol. 113, no. 2, pp. 748–753, 2008.

[12] T. Eren, "A note on minimizing maximum lateness in an *m*-machine scheduling problem with a learning effect," *Applied Mathematics and Computation*, vol. 209, no. 2, pp. 186–190, 2009.

[13] C.-C. Wu and W.-C. Lee, "A note on the total completion time problem in a permutation flowshop with a learning effect," *European Journal of Operational Research*, vol. 192, no. 1, pp. 343–347, 2009.

[14] R. Rudek, "Computational complexity and solution algorithms for flowshop scheduling problems with the learning effect," *Computers & Industrial Engineering*, vol. 61, no. 1, pp. 20–31, 2011.

[15] W.-H. Kuo, C.-J. Hsu, and D.-L. Yang, "Worst-case and numerical analysis of heuristic algorithms for flowshop scheduling problems with a time-dependent learning effect," *Information Sciences*, vol. 184, no. 1, pp. 282–297, 2012.

[16] W.-C. Lee and Y.-H. Chung, "Permutation flowshop scheduling to minimize the total tardiness with learning effects," *International Journal of Production Economics*, vol. 141, no. 1, pp. 327–334, 2013.

[17] L.-H. Sun, K. Cui, J.-H. Chen, J. Wang, and X.-C. He, "Research on permutation flow shop scheduling problems with general position-dependent learning effects," *Annals of Operations Research*, vol. 211, no. 1, pp. 473–480, 2013.

[18] L.-H. Sun, K. Cui, J.-H. Chen, J. Wang, and X.-C. He, "Some results of the worst-case analysis for flow shop scheduling with a learning effect," *Annals of Operations Research*, vol. 211, no. 1, pp. 481–490, 2013.

[19] T. C. E. Cheng, C.-C. Wu, J.-C. Chen, W.-H. Wu, and S.-R. Cheng, "Two-machine flowshop scheduling with a truncated learning function to minimize the makespan," *International Journal of Production Economics*, vol. 141, no. 1, pp. 79–86, 2013.

[20] G. Li, X.-Y. Wang, J.-B. Wang, and L.-Y. Sun, "Worst case analysis of flow shop scheduling problems with a time-dependent learning effect," *International Journal of Production Economics*, vol. 142, no. 1, pp. 98–104, 2013.

[21] J.-J. Wang and B.-H. Zhang, "Permutation flowshop problems with bi-criterion makespan and total completion time objective and position-weighted learning effects," *Computers & Operations Research*, vol. 58, pp. 24–31, 2015.

[22] J.-B. Wang and J.-J. Wang, "Flowshop scheduling with a general exponential learning effect," *Computers & Operations Research*, vol. 43, pp. 292–308, 2014.

[23] K. Lai, P.-H. Hsu, P.-H. Ting, and C.-C. Wu, "A truncated sum of processing-times-based learning model for a two-machine flowshop scheduling problem," *Human Factors and Ergonomics in Manufacturing & Service Industries*, vol. 24, no. 2, pp. 152–160, 2014.

[24] Y. Liu and Z. Feng, "Two-machine no-wait flowshop scheduling with learning effect and convex resource-dependent processing times," *Computers & Industrial Engineering*, vol. 75, pp. 170–175, 2014.

[25] W.-H. Wu, W. H. Wu, J. C. Chen, W. C. Lin, J. Wu, and C. C. Wu, "A heuristic-based genetic algorithm for the two-machine flowshop scheduling with learning consideration," *Journal of Manufacturing Systems*, vol. 35, pp. 223–233, 2015.

[26] Y.-R. Shiau, M.-S. Tsai, W.-C. Lee, and T. C. E. Cheng, "Two-agent two-machine flowshop scheduling with learning effects to minimize the total completion time," *Computers & Industrial Engineering*, vol. 87, pp. 580–589, 2015.

[27] Y.-Y. Lu, "Research on no-idle permutation flowshop scheduling with time-dependent learning effect and deteriorating jobs," *Applied Mathematical Modelling*, vol. 40, no. 4, pp. 3447–3450, 2016.

[28] H. Qin, Z.-H. Zhang, and D. Bai, "Permutation flowshop group scheduling with position-based learning effect," *Computers & Industrial Engineering*, vol. 92, pp. 1–15, 2016.

[29] H. He, "Minimization of maximum lateness in an *m*-machine permutation flow shop with a general exponential learning effect," *Computers & Industrial Engineering*, vol. 97, pp. 73–83, 2016.

[30] J.-B. Wang, F. Liu, and J.-J. Wang, "Research on *m*-machine flow shop scheduling with truncated learning effects," *International Transactions in Operational Research*, 2016.

[31] X.-Y. Wang, Z. Zhou, X. Zhang, P. Ji, and J.-B. Wang, "Several flow shop scheduling problems with truncated position-based learning effect," *Computers & Operations Research*, vol. 40, no. 12, pp. 2906–2929, 2013.

[32] C. Chu, "Efficient heuristics to minimize total flow time with release dates," *Operations Research Letters*, vol. 12, no. 5, pp. 321–330, 1992.

[33] M. I. Dessouky and J. S. Deogun, "Sequencing jobs with unequal ready times to minimize mean flow time," *SIAM Journal on Computing*, vol. 10, no. 1, pp. 192–202, 1981.

[34] C.-N. Potts, "Analysis of heuristics for two-machine flow-shop sequencing subject to release dates," *Mathematics of Operations Research*, vol. 10, no. 4, pp. 576–584, 1985.

[35] Y. Yin, W.-H. Wu, W.-H. Wu, and C.-C. Wu, "A branch-and-bound algorithm for a single machine sequencing to minimize the total tardiness with arbitrary release dates and position-dependent learning effects," *Information Sciences*, vol. 256, pp. 91–108, 2014.

[36] D. Bai, M. Tang, Z.-H. Zhang, and E. D. R. Santibanez-Gonzalez, "Flow shop learning effect scheduling problem with release dates," *Omega*, vol. 78, pp. 21–38, 2018.

[37] W.-C. Lee and C.-C. Wu, "Minimizing total completion time in a two-machine flowshop with a learning effect," *International Journal of Production Economics*, vol. 88, no. 1, pp. 85–93, 2004.

[38] W.-C. Lee, C.-C. Wu, and H.-J. Sung, "A bi-criterion single-machine scheduling problem with learning considerations," *Acta Informatica*, vol. 40, no. 4, pp. 303–315, 2004.

[39] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. R. Kan, "Optimization and approximation in deterministic sequencing and scheduling: a survey," *Annals of Discrete Mathematics*, vol. 5, pp. 287–326, 1979.

[40] J. M. Framinan and R. Leisten, "An efficient constructive heuristic for flowtime minimisation in permutation flow shops," *Omega*, vol. 31, no. 4, pp. 311–317, 2003.

[41] B. Shi, B. Meng, H. Yang, J. Wang, and W. Shi, "A novel approach for reducing attributes and its application to small enterprise financing ability evaluation," *Complexity*, vol. 2018, Article ID 1032643, 17 pages, 2018.

[42] Y. Zhou, L. Zhou, Y. Wang, Z. Yang, and J. Wu, "Application of multiple-population genetic algorithm in optimizing the train-set circulation plan problem," *Complexity*, vol. 2017, Article ID 3717654, 14 pages, 2017.

[43] Y. Yu, H. Ma, M. Yu, S. Ye, and X. Chen, "Multi-population management in evolutionary algorithms and application to complex warehouse scheduling problems," *Complexity*, vol. 2018, Article ID 4730957, 14 pages, 2018.

[44] F. Liu, S. Wang, Y. Hong, and X. Yue, "On the robust and stable flowshop scheduling under stochastic and dynamic disruptions," *IEEE Transactions on Engineering Management*, vol. 64, no. 4, pp. 539–553, 2017.

[45] H. Ishibuchi, T. Yoshida, and T. Murata, "Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 204–223, 2003.