Optim Lett (2012) 6:1393–1409
DOI 10.1007/s11590-011-0334-y

ORIGINAL PAPER

Minimizing makespan in a two-machine flow shop with effects of deterioration and learning

Ji-Bo Wang · P. Ji · T. C. E. Cheng · Dan Wang

Received: 6 September 2010 / Accepted: 25 April 2011 / Published online: 4 May 2011
© Springer-Verlag 2011

Abstract We consider a two-machine flow shop scheduling problem with effects of deterioration and learning. By the effects of deterioration and learning, we mean that the processing time of a job is a function of its execution starting time and its position in a sequence. The objective is to find a sequence that minimizes the makespan. Several dominance properties and two lower bounds are derived, which are used to speed up the elimination process of a branch-and-bound algorithm proposed to solve the problem. Two heuristic algorithms are also proposed to obtain near-optimal solutions. Computational results are presented to evaluate the performance of the proposed algorithms.

Keywords Scheduling · Flow shop · Deteriorating jobs · Learning effect · Makespan · Branch-and-bound algorithm

1 Introduction

In many branches of industry and logistics, there arise problems of ordering jobs on machines (Pardalos [19], Pardalos and Resende [18], Setamaa-Karkkainen et al. [20],

J.-B. Wang (✉) · D. Wang
School of Science, Shenyang Aerospace University, Shenyang 110136, China
e-mail: wangjibo75@yahoo.com.cn

P. Ji
Department of Industrial and Systems Engineering, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong

T. C. E. Cheng
Department of Logistics and Maritime Studies, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong

Lee and Yu [14], Hadda [9], Gawiejnowicz [8], and Floudas and Pardalos [7]). In classical scheduling theory, the job processing times are considered to be constant. In practice, however, we often encounter situations in which the job processing times may be subject to change due to the phenomena of learning and/or deterioration. Job deterioration appears, e.g., in scheduling maintenance jobs or cleaning assignments, where any delay in processing a job is penalized by incurring additional time for accomplishing the job. Job learning effect appears when the processing times arise from manual operations, in which case the possibility of learning exists.

Extensive surveys of scheduling models and problems involving jobs with deteriorating jobs can be found in Alidaee and Womer [1], Cheng et al. [4] and Gawiejnowicz [8]. An extensive review of research on scheduling with learning effect and/or deteriorating jobs could be found in Biskup [3]. More recent papers that consider scheduling jobs with learning effect and/or deteriorating jobs include Wu et al. [29], Shiau et al. [21], Toksar and Guner [22], Wu and Lee [27,28], Cheng et al. [5,6], Lee and Wu [12], Lee et al. [13], Wang and Liu [24], and Ng et al. [17], Zhao and Tang [30], Wang and Wang [25], and Wang et al. [26].

On the other hand, the phenomenon that learning effect and deteriorating jobs occur simultaneously can be observed in real-life situations. For example, as the manufacturing environment becomes increasingly competitive, in order to provide customers with greater product variety, organizations are moving toward shorter production runs and frequent product changes. The learning and forgetting that workers undergo in this environment have thus become increasingly important as workers tend to spend more time in rotating among tasks and responsibilities prior to becoming fully proficient. These workers are often interrupted by product and process changes, which inadvertently affect performance. For simplicity, we refer to this as forgetting. Considering learning and forgetting effects in measuring productivity should be helpful in improving the accuracy of production planning and productivity estimation (Nembhard and Osothsilp [16]). Another practical example that motivates the above scheduling model is the manual production of glass crafts by a skilled craftsman. Silicon-based raw material is first heated up in an oven until it becomes a lump of malleable dough from which the craftsman cuts pieces and shapes them according to different designs into different glass craft products. The initial time to heat up the raw material to the threshold temperature at which it can be shaped is long and so the first piece (i.e., job) has a long processing time, which includes both the heating time (i.e., the deterioration effect) and the shaping time (i.e., the normal processing time). The second piece requires a shorter time to re-heat the dough to the threshold temperature (i.e., a smaller deterioration effect). Similarly, the later a piece is cut from the dough, the shorter is its heating time to reach the threshold temperature. On the other hand, the pieces that are shaped later require shorter shaping times because the craftsman's productivity improves as a result of learning (Cheng et al. [6]). This paper extends the results of Lee [10] and Wang and Liu [24] by considering a two-machine flow shop scheduling with the effects of deterioration and learning to minimize makespan. We provide the exact solution and near-optimal solutions for the two-machine flow shop makespan scheduling problem with the effects of deterioration and learning.

The rest of this paper is organized as follows. In the next section we give the problem description. In Sect. 3 we propose several elimination rules, which can be used to

enhance the search for the optimal solution. In Sect. 4 we establish an exact algorithm and two heuristics to search for the optimal and near-optimal solutions. In Sect. 5 we present computational experiments to evaluate the performance of the branch-and-bound algorithm and the heuristic algorithms. Concluding remarks are given in the last section.

2 Problem description

Two-machine flow shop scheduling problems are frequently found in industry and are characterized by a set of jobs, $N = \{J_1, J_2, \ldots, J_n\}$ and a set of the two machines $M = \{M_1, M_2\}$. The set of n jobs are to be processed sequentially on the two machines. That is, the jobs follow the same permutation, starting with machine M_1, followed by machine M_2. Let $p_{ijr}(t)$ be the processing time of job J_j on machine M_i if it is started at time t and scheduled in position r in a sequence. From Lee [10] and Wang and Liu [24], we consider the following model:

$$p_{ijr}(t) = \alpha_{ij} t r^a, \tag{1}$$

where α_{ij} denotes the deterioration rate of job J_j on machine M_i and $a \leq 0$ denotes a constant learning index (Biskup [2]). All the jobs are available for processing at time $t_0 > 0$. The objective is to find a schedule that minimizes the makespan, i.e., the maximum completion time of all the jobs. We assume unlimited intermediate storage between successive machines for the general flow shop scheduling problem.

Let $C_{i,j}(\pi)$ denote the completion time of job J_j on machine M_i under schedule π. Let $C_{i,[j]}(\pi)$ denote the completion time of the jth job on machine M_i under schedule π. Thus the completion time of job J_j is $C_j = C_{2,j}$. Using the three-field notation for scheduling problem classification, the problem can be denoted as $F2|p_{ijr}(t) = \alpha_{ij} t r^a|C_{\max}$ (the complexity of this problem is still open). For ease of exposition, we denote p_{1j} by α_j, and p_{2j} by β_j, $j = 1, 2, \ldots, n$. Since unlimited intermediate storage is assumed, it is evident that an optimal schedule exists with no idle time between consecutive jobs on machine M_1. For a given scheduling $\pi = [J_1, J_2, \ldots, J_n]$, the completion time of the rth job on machine M_1 is given by

$$C_{1,[r]} = t_0 \prod_{i=1}^{r} \left(1 + \alpha_i i^a\right), \quad r = 1, 2, \ldots, n. \tag{2}$$

3 Dominance properties

Let S_1 and S_2 be two job schedules in which the difference between S_1 and S_2 is a pairwise interchange of two adjacent jobs J_i and J_j, i.e., $S_1 = (\pi, J_i, J_j, \pi')$ and $S_2 = (\pi, J_j, J_i, \pi')$, where π and π' are partial sequences. Further, we assume that there are $r-1$ jobs in π. Thus, J_i and J_j are the rth and the $(r+1)$th jobs, respectively. Meanwhile, J_j and J_i are scheduled in the rth and the $(r+1)$th positions in S_2. To

further simplify notation, let A and B denote the completion times of the last job in π on M_1 and M_2, respectively.

It is clear that the completion times of J_k in S_1 and S_2 are equal if it is in π since it is in the same position in both sequences. To show S_1 dominates S_2, it suffices to show that the $(r+1)$th jobs in S_1 and S_2 satisfy the condition

$$C_{[r+1]}(S_1) \le C_{[r+1]}(S_2), \tag{3}$$

where [ ] denotes the position of a job in a sequence. Three separate cases need to be considered: (i) $\frac{B-A}{Ar^a} \le \alpha_i \le \alpha_j$, (ii) $\alpha_i \le \frac{B-A}{Ar^a} \le \alpha_j$, and (iii) $\alpha_i \le \alpha_j \le \frac{B-A}{Ar^a}$; each case is divided into 4 subcases.

Proposition 1.1 *If* $\frac{B-A}{Ar^a} \le \alpha_i \le \alpha_j, \alpha_i \ge \left(\frac{r}{r+1}\right)^a \max\{\beta_i, \beta_j\}$ *and* $(1+\alpha_i r^a)(1+\alpha_j(r+1)^a)(1+\beta_j(r+1)^a) \le (1+\alpha_j r^a)(1+\alpha_i(r+1)^a)(1+\beta_i(r+1)^a)$, *then* S_1 *dominates* S_2.

Proof Since $\frac{B-A}{Ar^a} \le \alpha_i \le \alpha_j$ and $\alpha_i \ge \left(\frac{r}{r+1}\right)^a \max\{\beta_i, \beta_j\}$, the completion times of J_i and J_j in S_1 and S_2 are

$$\begin{aligned}
C_{[r]}(S_1) &= \max\left\{A(1+\alpha_i r^a), B\right\}\left(1+\beta_i r^a\right) = A\left(1+\alpha_i r^a\right)\left(1+\beta_i r^a\right),\\
C_{[r+1]}(S_1) &= \max\left\{A(1+\alpha_i r^a)(1+\alpha_j(r+1)^a), A\left(1+\alpha_i r^a\right)\left(1+\beta_i r^a\right)\right\}\\
&\quad\times\left(1+\beta_j(r+1)^a\right)\\
&= A\left(1+\alpha_i r^a\right)\left(1+\alpha_j(r+1)^a\right)\left(1+\beta_j(r+1)^a\right),\\
C_{[r]}(S_2) &= \max\left\{A(1+\alpha_j r^a), B\right\}\left(1+\beta_j r^a\right) = A\left(1+\alpha_j r^a\right)\left(1+\beta_j r^a\right),
\end{aligned}$$

and

$$\begin{aligned}
C_{[r+1]}(S_2) &= \max\left\{A(1+\alpha_j r^a)(1+\alpha_i(r+1)^a), A(1+\alpha_j r^a)(1+\beta_j r^a)\right\}\\
&\quad\times\left(1+\beta_i(r+1)^a\right)\\
&= A\left(1+\alpha_j r^a\right)\left(1+\alpha_i(r+1)^a\right)\left(1+\beta_i(r+1)^a\right).
\end{aligned}$$

Thus, from $(1+\alpha_i r^a)\left(1+\alpha_j(r+1)^a\right)\left(1+\beta_j(r+1)^a\right) \le \left(1+\alpha_j r^a\right)(1+\alpha_i(r+1)^a)(1+\beta_i(r+1)^a)$, we have

$$C_{[r+1]}(S_1) \le C_{[r+1]}(S_2).$$

Therefore, S_1 dominates S_2. □

Proposition 1.2 *If* $\frac{B-A}{Ar^a} \le \alpha_i \le \alpha_j, \beta_i \ge \alpha_j\left(\frac{r+1}{r}\right)^a, \beta_j \ge \alpha_i\left(\frac{r+1}{r}\right)^a$ *and* $(1+\alpha_i r^a)(1+\beta_i r^a)(1+\beta_j(r+1)^a) \le (1+\alpha_j r^a)(1+\beta_j r^a)(1+\beta_i(r+1)^a)$, *then* S_1 *dominates* S_2.

Proposition 1.3 *If* $\frac{B-A}{Ar^a} \le \alpha_i \le \alpha_j, \beta_i \ge \alpha_j\left(\frac{r+1}{r}\right)^a, \beta_j \le \alpha_i\left(\frac{r+1}{r}\right)^a$ *and* $(1+\alpha_i r^a)(1+\beta_i r^a)(1+\beta_j(r+1)^a) \le (1+\alpha_j r^a)(1+\alpha_i(r+1)^a)(1+\beta_i(r+1)^a)$, *then* S_1 *dominates* S_2.

Proposition 1.4 *If* $\frac{B-A}{Ar^a} \leq \alpha_i \leq \alpha_j, \beta_i \leq \alpha_j \left(\frac{r+1}{r}\right)^a, \beta_j \geq \alpha_i \left(\frac{r+1}{r}\right)^a$ *and* $(1+\alpha_i r^a)(1+\alpha_j(r+1)^a)(1+\beta_j(r+1)^a) \leq (1+\alpha_j r^a)(1+\beta_j r^a)(1+\beta_i(r+1)^a)$, *then* S_1 *dominates* S_2.

Proposition 2.1 *If* $\alpha_i \leq \frac{B-A}{Ar^a} \leq \alpha_j$, $A\left(1+\alpha_i r^a\right)\left(1+\alpha_j(r+1)^a\right) \leq B\left(1+\beta_i r^a\right)$, $\alpha_i \leq \beta_j \left(\frac{r}{r+1}\right)^a$ *and* $B\left(1+\beta_i r^a\right)\left(1+\beta_j(r+1)^a\right) \leq A\left(1+\alpha_j r^a\right)\left(1+\beta_j r^a\right)(1+\beta_i(r+1)^a)$, *then* S_1 *dominates* S_2.

Proposition 2.2 *If* $\alpha_i \leq \frac{B-A}{Ar^a} \leq \alpha_j$, $A\left(1+\alpha_i r^a\right)\left(1+\alpha_j(r+1)^a\right) \geq B\left(1+\beta_i r^a\right)$, $\alpha_i \geq \beta_j \left(\frac{r}{r+1}\right)^a$ *and* $\left(1+\alpha_i r^a\right)\left(1+\alpha_j(r+1)^a\right)\left(1+\beta_j(r+1)^a\right) \leq \left(1+\alpha_j r^a\right)(1+\alpha_i(r+1)^a)\left(1+\beta_i(r+1)^a\right)$, *then* S_1 *dominates* S_2.

Proposition 2.3 *If* $\alpha_i \leq \frac{B-A}{Ar^a} \leq \alpha_j$, $A\left(1+\alpha_i r^a\right)\left(1+\alpha_j(r+1)^a\right) \leq B\left(1+\beta_i r^a\right)$, $\alpha_i \geq \beta_j \left(\frac{r}{r+1}\right)^a$ *and* $B\left(1+\beta_i r^a\right)\left(1+\beta_j(r+1)^a\right) \leq A\left(1+\alpha_j r^a\right)\left(1+\alpha_i(r+1)^a\right)\left(1+\beta_i(r+1)^a\right)$, *then* S_1 *dominates* S_2.

Proposition 2.4 *If* $\alpha_i \leq \frac{B-A}{Ar^a} \leq \alpha_j$, $A\left(1+\alpha_i r^a\right)\left(1+\alpha_j(r+1)^a\right) \geq B\left(1+\beta_i r^a\right)$, $\alpha_i \leq \beta_j \left(\frac{r}{r+1}\right)^a$ *and* $\left(1+\alpha_i r^a\right)\left(1+\alpha_j(r+1)^a\right)\left(1+\beta_j(r+1)^a\right) \leq \left(1+\alpha_j r^a\right)\left(1+\beta_j r^a\right)\left(1+\beta_i(r+1)^a\right)$, *then* S_1 *dominates* S_2.

Proposition 3.1 *If* $\alpha_i \leq \alpha_j \leq \frac{B-A}{Ar^a}$, $A\left(1+\alpha_i r^a\right)\left(1+\alpha_j(r+1)^a\right) \leq B\left(1+\beta_i r^a\right)$, $A\left(1+\alpha_j r^a\right)\left(1+\alpha_i(r+1)^a\right) \leq B\left(1+\beta_j r^a\right)$ *and* $\beta_i \leq \beta_j$, *then* S_1 *dominates* S_2.

Proposition 3.2 *If* $\alpha_i \leq \alpha_j \leq \frac{B-A}{Ar^a}$, $A\left(1+\alpha_i r^a\right)\left(1+\alpha_j(r+1)^a\right) \geq B\left(1+\beta_i r^a\right)$ $A\left(1+\alpha_j r^a\right)\left(1+\alpha_i(r+1)^a\right) \geq B(1+\beta_j r^a)$ *and* $\left(1+\alpha_i r^a\right)\left(1+\alpha_j(r+1)^a\right)\left(1+\beta_j(r+1)^a\right) \leq (1+\alpha_j r^a)(1+\alpha_i(r+1)^a)\left(1+\beta_i(r+1)^a\right)$, *then* S_1 *dominates* S_2.

Proposition 3.3 *If* $\alpha_i \leq \alpha_j \leq \frac{B-A}{Ar^a}$, $A(1+\alpha_i r^a)(1+\alpha_j(r+1)^a) \leq B(1+\beta_i r^a)$, $A(1+\alpha_j r^a)(1+\alpha_i(r+1)^a) \geq B(1+\beta_j r^a)$ *and* $B(1+\beta_i r^a)(1+\beta_j(r+1)^a) \leq A(1+\alpha_j r^a)(1+\alpha_i(r+1)^a)(1+\beta_i(r+1)^a)$, *then* S_1 *dominates* S_2.

Proposition 3.4 *If* $\alpha_i \leq \alpha_j \leq \frac{B-A}{Ar^a}$, $A(1+\alpha_i r^a)(1+\alpha_j(r+1)^a) \geq B(1+\beta_i r^a)$, $A(1+\alpha_j r^a)(1+\alpha_i(r+1)^a) \leq B(1+\beta_j r^a)$ *and* $A(1+\alpha_i r^a)(1+\beta_j(r+1)^a)(1+\beta_j(r+1)^a) \leq B(1+\beta_j r^a)(1+\beta_i(r+1)^a)$, *then* S_1 *dominates* S_2.

To further curtail the size of a branching tree, we develop two dominance properties for a pairwise interchange of two non-adjacent jobs. Let S_1 and S_2 be two job schedules in which the difference between S_1 and S_2 is a pairwise interchange of two non-adjacent jobs J_i and J_j, i.e., $S_1 = (\pi, J_i, \pi', J_j, \pi'')$ and $S_2 = (\pi, J_j, \pi', J_i, \pi'')$, where π, π' and π'' are partial sequences, and J_i and J_j are in positions r and τ in schedule S_1.

Theorem 1 *If jobs* J_i *and* J_j *satisfy* $\alpha_i > \alpha_j$ *and* $\beta_i = \beta_j$, *then* S_2 *dominates* S_1.

Proof It should be clear that the completion times of J_k in sequences S_1 and S_2 are equal if it is in π because both sequences have the same jobs in these

positions, i.e., $C_{1[r-1]}(S_2) = C_{1[r-1]}(S_2)$, $C_{2[r-1]}(S_2) = C_{2[r-1]}(S_2)$ for positions $k = 1, 2, \ldots, r-1$.

The completion times of J_i and J_j in S_2 are, respectively,

$$\begin{aligned} C_{2[r]}(S_2) &= \max\left\{C_{1[r-1]}(S_2) + \alpha_j C_{1[r-1]}(S_2) r^a, C_{2[r-1]}(S_2)\right\} \\ &\quad + \beta \max\left\{C_{1[r-1]}(S_2) + \alpha_j C_{1[r-1]}(S_2) r^a, C_{2[r-1]}(S_2)\right\} r^a \\ &= \max\left\{C_{1[r-1]}(S_2)(1+\alpha_j r^a)(1+\beta r^a), C_{2[r-1]}(S_2)(1+\beta r^a)\right\}, \\ C_{2[r]}(S_1) &= \max\left\{C_{1[r-1]}(S_2)(1+\alpha_i r^a)(1+\beta r^a), C_{2[r-1]}(S_2)(1+\beta r^a)\right\}, \end{aligned}$$

From these two equations, we have

$$C_{2[r]}(S_2) \le C_{2[r]}(S_1) \tag{4}$$

since $\alpha_i \ge \alpha_j$.

For positions $k = r+1, r = 2, \ldots, \tau - 1$,

$$\begin{aligned} C_{2[k]}(S_2) = \max\Bigg\{ & C_{1[r-1]}(S_2)(1+\alpha_j r^a) \prod_{l=1}^{k} (1+\alpha_{[l]}(r+l)^a)(1+\beta r^{\tau-1}), \\ & C_{1[r-1]}(S_2)(1+\alpha_j r^a) \prod_{l=1}^{k-1} (1+\alpha_{[l]}(r+l)^a)(1+\beta r^{\tau-2})(1+\beta r^{\tau-1}), \\ & \ldots \\ & C_{1[r-1]}(S_2)(1+\alpha_j r^a) \prod_{l=r}^{\tau-1} (1+\beta l^a), \\ & C_{2[r-1]}(S_2)(1+\beta r^{\tau-1}) \Bigg\}, \\ C_{2[k]}(S_1) = \max\Bigg\{ & C_{1[r-1]}(S_2)(1+\alpha_i r^a) \prod_{l=1}^{k} (1+\alpha_{[l]}(r+l)^a)(1+\beta r^{\tau-1}), \\ & C_{1[r-1]}(S_2)(1+\alpha_i r^a) \prod_{l=1}^{k-1} (1+\alpha_{[l]}(r+l)^a)(1+\beta r^{\tau-2})(1+\beta r^{\tau-1}), \\ & \ldots \\ & C_{1[r-1]}(S_2)(1+\alpha_i r^a) \prod_{l=r}^{\tau-1} (1+\beta l^a), \\ & C_{2[r-1]}(S_2)(1+\beta r^{\tau-1}) \Bigg\}. \end{aligned}$$

Since both equations have the same positions except for position τ, we have

$$C_{2[k]}(S_2) \leq C_{2[k]}(S_1) \tag{5}$$

since $\alpha_i \geq \alpha_j$.

For position $k = \tau$,

$$\begin{aligned}
C_{2[\tau]}(S_2) = \max \Bigg\{ & C_{1[r-1]}(S_2)(1+\alpha_j r^a)(1+\alpha_i \tau^a) \prod_{l=r+1}^{\tau-1} (1+\alpha_{[l]} l^a)(1+\beta \tau^a), \\
& C_{1[r-1]}(S_2)(1+\alpha_j r^a) \prod_{l=r+1}^{\tau-1} (1+\alpha_{[l]} l^a)(1+\beta r^{\tau-1})(1+\beta r^{\tau}), \\
& C_{1[r-1]}(S_2)(1+\alpha_j r^a) \prod_{l=r+1}^{\tau-2} (1+\alpha_{[l]} l^a)(1+\beta r^{\tau-2})(1+\beta r^{\tau-1})(1+\beta r^{\tau}), \\
& \cdots \\
& C_{1[r-1]}(S_2)(1+\alpha_j r^a) \prod_{l=r}^{\tau} (1+\beta l^a), \\
& C_{2[r-1]}(S_2)(1+\beta r^{\tau}) \Bigg\},
\end{aligned}$$

$$\begin{aligned}
C_{2[\tau]}(S_1) = \max \Bigg\{ & C_{1[r-1]}(S_2)(1+\alpha_i r^a)(1+\alpha_j \tau^a) \prod_{l=r+1}^{\tau-1} (1+\alpha_{[l]} l^a)(1+\beta \tau^a), \\
& C_{1[r-1]}(S_2)(1+\alpha_i r^a) \prod_{l=r+1}^{\tau-1} (1+\alpha_{[l]} l^a)(1+\beta r^{\tau-1})(1+\beta r^{\tau}), \\
& C_{1[r-1]}(S_2)(1+\alpha_i r^a) \prod_{l=r+1}^{\tau-2} (1+\alpha_{[l]} l^a)(1+\beta r^{\tau-2})(1+\beta r^{\tau-1})(1+\beta r^{\tau}), \\
& \cdots \\
& C_{1[r-1]}(S_2)(1+\alpha_i r^a) \prod_{l=r}^{\tau} (1+\beta l^a), \\
& C_{2[r-1]}(S_2)(1+\beta r^{\tau}) \Bigg\}.
\end{aligned}$$

From these two equations, we have

$$C_{2[\tau]}(S_2) \leq C_{2[\tau]}(S_1) \tag{6}$$

since $\alpha_i \geq \alpha_j$ and $(1+\alpha_j r^a)(1+\alpha_i \tau^a) \leq (1+\alpha_i r^a)(1+\alpha_j \tau^a)$.

For the jobs that are in π', their completion times may be delayed due to (6), i.e.,

$$C_{[k]}(S_2) \le C_{[k]}(S_1), \quad \text{if } J_{[k]} \in \pi'. \tag{7}$$

Hence, S_2 dominates S_1. □

Theorem 2 *If jobs J_i and J_j satisfy $\alpha_i = \beta_i, \alpha_j = \beta_j$, and $\alpha_i > \alpha_j$, then S_2 dominates S_1.*

Proof It should be clear that the completion times of J_k in sequences S_1 and S_2 are equal if it is in π because both sequences have the same jobs in these positions, i.e., $C_{1[r-1]}(S_2) = C_{1[r-1]}(S_2)$, $C_{2[r-1]}(S_2) = C_{2[r-1]}(S_2)$ for positions $k = 1, 2, \ldots, r-1$.

The completion times of J_i and J_j in S_2 are, respectively,

$$\begin{aligned}
C_{2[r]}(S_2) &= \max\left\{C_{1[r-1]}(S_2) + \alpha_j C_{1[r-1]}(S_2) r^a, C_{2[r-1]}(S_2)\right\} \\
&\quad + \alpha_j \max\left\{C_{1[r-1]}(S_2) + \alpha_j C_{1[r-1]}(S_2) r^a, C_{2[r-1]}(S_2)\right\} r^a \\
&= \max\left\{C_{1[r-1]}(S_2)(1+\alpha_j r^a)^2, (C_{2[r-1]}(S_2)(1+\alpha_j r^a)\right\}, \\
C_{2[r]}(S_1) &= \max\left\{C_{1[r-1]}(S_2)(1+\alpha_i r^a)^2, (C_{2[r-1]}(S_2)(1+\alpha_i r^a)\right\},
\end{aligned}$$

From these two equations, we have

$$C_{2[r]}(S_2) \le C_{2[r]}(S_1) \tag{8}$$

since $\alpha_i \ge \alpha_j$.

For positions $k = r+1, r = 2, \ldots, \tau - 1$,

$$\begin{aligned}
C_{2[k]}(S_2) = \max\Bigg\{ & C_{1[r-1]}(S_2)(1+\alpha_j r^a) \prod_{l=r+1}^{k} (1+\alpha_{[l]} l^a)(1+\alpha_{[k]} k^a), \\
& C_{1[r-1]}(S_2)(1+\alpha_j r^a) \prod_{l=r+1}^{k-1} (1+\alpha_{[l]} l^a) \prod_{l=k-1}^{k} (1+\alpha_{[l]} l^a), \\
& C_{1[r-1]}(S_2)(1+\alpha_j r^a) \prod_{l=r+1}^{k-2} (1+\alpha_{[l]} l^a) \prod_{l=k-2}^{k} (1+\alpha_{[l]} l^a), \\
& \cdots \\
& C_{1[r-1]}(S_2)(1+\alpha_j r^a)^2 \prod_{l=r+1}^{k} (1+\alpha_{[l]} l^a), \\
& C_{2[r-1]}(S_2)(1+\alpha_j r^a) \prod_{l=r+1}^{k} (1+\alpha_{[l]} l^a) \Bigg\}, \\
C_{2[k]}(S_1) = \max\Bigg\{ & C_{1[r-1]}(S_2)(1+\alpha_i r^a) \prod_{l=r+1}^{k} (1+\alpha_{[l]} l^a)(1+\alpha_{[k]} k^a),
\end{aligned}$$

$$C_{1[r-1]}(S_2)(1+\alpha_i r^a)\prod_{l=r+1}^{k-1}(1+\alpha_{[l]}l^a)\prod_{l=k-1}^{k}(1+\alpha_{[l]}l^a),$$
$$C_{1[r-1]}(S_2)(1+\alpha_i r^a)\prod_{l=r+1}^{k-2}(1+\alpha_{[l]}l^a)\prod_{l=k-2}^{k}(1+\alpha_{[l]}l^a),$$
$$\cdots$$
$$C_{1[r-1]}(S_2)(1+\alpha_i r^a)^2\prod_{l=r+1}^{k}(1+\alpha_{[l]}l^a),$$
$$C_{2[r-1]}(S_2)(1+\alpha_i r^a)\prod_{l=r+1}^{k}(1+\alpha_{[l]}l^a)\Bigg\}.$$

Since both equations have the same positions except for position τ, we have

$$C_{2[k]}(S_2) \le C_{2[k]}(S_1) \tag{9}$$

since $\alpha_i \ge \alpha_j$.

For position $k = \tau$,

$$C_{2[\tau]}(S_2) = \max\Bigg\{C_{1[r-1]}(S_2)(1+\alpha_j r^a)\prod_{l=r+1}^{\tau-1}(1+\alpha_{[l]}l^a)(1+\alpha_i\tau^a)^2,$$
$$C_{1[r-1]}(S_2)(1+\alpha_j r^a)\prod_{l=r+1}^{\tau-2}(1+\alpha_{[l]}l^a)\prod_{l=\tau-2}^{\tau-1}(1+\alpha_{[l]}l^a)(1+\alpha_i\tau^a),$$
$$C_{1[r-1]}(S_2)(1+\alpha_j r^a)\prod_{l=r+1}^{\tau-3}(1+\alpha_{[l]}l^a)\prod_{l=\tau-3}^{\tau-1}(1+\alpha_{[l]}l^a)(1+\alpha_i\tau^a),$$
$$\cdots$$
$$C_{1[r-1]}(S_2)(1+\alpha_j r^a)^2\prod_{l=r+1}^{\tau-1}(1+\alpha_{[l]}l^a)(1+\alpha_i\tau^a),$$
$$C_{2[r-1]}(S_2)(1+\alpha_j r^a)\prod_{l=r+1}^{\tau-1}(1+\alpha_{[l]}l^a)(1+\alpha_i\tau^a)\Bigg\},$$

$$C_{2[\tau]}(S_1) = \max\Bigg\{C_{1[r-1]}(S_2)(1+\alpha_i r^a)\prod_{l=r+1}^{\tau-1}(1+\alpha_{[l]}l^a)(1+\alpha_j\tau^a)^2,$$
$$C_{1[r-1]}(S_2)(1+\alpha_i r^a)\prod_{l=r+1}^{\tau-2}(1+\alpha_{[l]}l^a)\prod_{l=\tau-2}^{\tau-1}(1+\alpha_{[l]}l^a)(1+\alpha_j\tau^a),$$
$$C_{1[r-1]}(S_2)(1+\alpha_i r^a)\prod_{l=r+1}^{\tau-3}(1+\alpha_{[l]}l^a)\prod_{l=\tau-3}^{\tau-1}(1+\alpha_{[l]}l^a)(1+\alpha_j\tau^a),$$
$$\cdots$$

$$C_{1[r-1]}(S_2)(1+\alpha_i r^a)^2 \prod_{l=r+1}^{\tau-1}(1+\alpha_{[l]}l^a)(1+\alpha_j\tau^a),$$
$$C_{2[r-1]}(S_2)(1+\alpha_i r^a) \prod_{l=r+1}^{\tau-1}(1+\alpha_{[l]}l^a)(1+\alpha_j\tau^a)\Bigg\}.$$

From these two equations, since $\alpha_i \geq \alpha_j$, we have $(1+\alpha_i r^a)^2(1+\alpha_j\tau^a) \geq (1+\alpha_j r^a)^2(1+\alpha_i\tau^a)$ and $(1+\alpha_i r^a)(1+\alpha_j\tau^a) \geq (1+\alpha_j r^a)(1+\alpha_i\tau^a)$, hence

$$C_{2[\tau]}(S_2) \leq C_{2[\tau]}(S_1). \tag{10}$$

For the jobs that are in π', their completion times may be delayed due to (10), i.e.,

$$C_{[k]}(S_2) \leq C_{[k]}(S_1), \quad \text{if } J_{[k]} \in \pi'. \tag{11}$$

Hence, S_2 dominates S_1. □

4 Algorithms

In this section we give two lower bounds to curtail the size of the branching tree and an initial upper bound by using two heuristic algorithms. We also construct a branch-and-bound algorithm to solve small-sized problems.

4.1 The heuristic algorithms

In this subsection two heuristic algorithms are presented. The procedure is adapted from the results in Wang et al. [23], Mosheiov [15], and Lee and Wu [11].

For the problem $F2|p_{ij}(t) = \alpha_{ij}t|C_{\max}$, Mosheiov [15] proposes an optimal algorithm (Algorithm SOLVE-FS2), but it is noted that this algorithm might not provide the optimal solution for this problem with a learning effect. Hence a heuristic algorithm is proposed, which is based on Algorithm SOLVE-FS2.

Wang et al. [23] propose three heuristic algorithms for the problem under study in this paper when there are no deteriorating jobs. The first method focuses on avoiding or reducing idle time on the second machine. The second method focuses on avoiding or reducing the waiting time of the jobs on the second machine. The third method is to choose the jobs that yield local optimality. Their computational results show that the first heuristic method is superior to the others. Lee and Wu [11] propose a two-phase heuristic algorithm for the same problem with a learning effect.

In summary, the steps of the procedures are as follows:

Heuristic Algorithm 1 (HA1)

Phase I

Step 1. Partition the jobs into two sets with set N_1 containing all the jobs with $\alpha_j \leq \beta_j$ and set N_2 all the jobs with $\alpha_j > \beta_j$.

Step 2. The jobs in N_1 go first in non-decreasing order of α_j and the jobs in N_2 follow in non-increasing order of β_j.

Phase II

Step 1. Let S_0 be the initial solution obtained from **Phase I**.

Step 2. Set $k = 1$ and $i = k + 1$.

Step 3. Create a new sequence S_1 by moving $J_{[i]}$ in S_0 forward to position k. Replace S_0 by S_1 if the value of the makespan of S_1 is smaller than that of S_0.

Step 4. If $i < n$, then set $i = i + 1$, go to Step 3.

Step 5. If $k < n - 1$, then set $k = k + 1$, go to Step 2. Otherwise, stop.

Heuristic Algorithm 2 (HA2)

Phase I

Step 1. Set $k = 1$ and $N = \{J_1, J_2, \ldots, J_n\}$.

Step 2. Choose the job J_i with the smallest $(1 + \alpha_i)(1 + \beta_i)$ to be scheduled in the kth position. Set $A_k = t_0(1 + \alpha_i)$ and $C_{[k]} = t_0(1 + \alpha_i)(1 + \beta_i)$. Delete J_i from N.

Step 3. Collect the jobs that satisfy the condition $A_k(1 + \alpha_i(k + 1)^a) \leq C_{[k]}$.

Step 3.1. If more than one job meets the condition, then choose the job J_i having the smallest α_i to be scheduled in the $(k + 1)$th position.

Step 3.2. If no job meets the condition, then choose the job J_i with the smallest processing time β_i on M_1 to be scheduled in the $(k+1)$th position. Update $C_{[k]}$ by $C_{[k+1]} = \max\{A_k(1 + \alpha_i(k + 1)^a), C_{[k]}\}(1 + \beta_i(k + 1)^a)$, and A_k by $A_{k+1} = A_k(1 + \alpha_i(k + 1)^a)$. Delete J_i from N. Increase k by 1.

Step 4. If N is not empty, then go to Step 3. Otherwise, Stop.

Phase II

Step 1. Let S_0 be the initial solution obtained from **Phase I**.

Step 2. Set $k = 1$ and $i = k + 1$.

Step 3. Create a new sequence S_1 by moving $J_{[i]}$ in S_0 forward to position k. Replace S_0 by S_1 if the value of the makespan of S_1 is smaller than that of S_0.

Step 4. If $i < n$, then set $i = i + 1$, go to Step 3.

Step 5. If $k < n - 1$, then set $k = k + 1$, go to Step 2. Otherwise, stop.

4.2 Lower bounds

The efficiency of a branch-and-bound algorithm largely depends on the use of effective lower bounds to trim the partial sequences. In the following we derive two lower

bounds to curtail the size of the branching tree. Suppose that π is a partial schedule in which the order of the first k jobs has been determined and S is a complete schedule obtained from π. By definition, the completion time of the $(k+1)$th job is

$$\begin{aligned} C_{[k+1]}(S) &= \max\left\{t_0\prod_{j=1}^{k+1}(1+\alpha_{[j]}j^a), C_{[k]}(S)\right\}(1+\beta_{[k+1]}(k+1)^a) \\ &\geq C_{[k]}(S)(1+\beta_{[k+1]}(k+1)^a). \end{aligned}$$

Similarly,

$$\begin{aligned} C_{[i]}(S) &= \max\left\{t_0\prod_{j=1}^{i}(1+\alpha_{[j]}j^a), C_{[i-1]}(S)\right\}(1+\beta_{[i]}i^a) \\ &\geq C_{[k]}(S)\prod_{j=k+1}^{i}(1+\beta_{[j]}j^a) \quad \text{for } i=k+1,\ldots,n. \end{aligned}$$

Therefore, the makespan of S is

$$C_{\max}(S) = C_{[n]}(S) \geq C_{[k]}(S)\prod_{j=k+1}^{n}(1+\beta_{[j]}j^a). \tag{12}$$

Observe that the first term on the right-hand side of Eq. 12 is known, the term $C_{[k]}(S)$ is known and a lower bound for π can be obtained by minimizing the term $\prod_{j=k+1}^{n}(1+\beta_{[i]}i^a)$. Lee [10] proves that the makespan is minimized by sequencing the jobs according to the shortest remaining deterioration rate (SRPT) rule. Thus, the minimal value of Eq. 12 can be obtained by sequencing the unscheduled jobs in non-decreasing order of the deterioration rates. Consequently, we obtain the first lower bound

$$LB_1 = C_{[k]}(S)\prod_{j=k+1}^{n}(1+\beta_{(j)}j^a), \tag{13}$$

where $\beta_{(k+1)} \leq \beta_{(k+2)} \leq \cdots \leq \beta_{(n)}$ is a non-decreasing order of the deterioration rates on M_2 for the remaining unscheduled jobs. However, if the deterioration rates on the first machine are large, this lower bound may not be tight. To overcome this situation, we need to take the deterioration rates on M_1 into consideration. In general, we have

$$\begin{aligned} C_{[i]} &= \max\left\{t_0\prod_{j=1}^{i}(1+\alpha_{[j]}j^a), C_{[i-1]}(S)\right\}(1+\beta_{[i]}i^a) \\ &\geq t_0\prod_{j=1}^{i}(1+\alpha_{[j]}j^a)(1+\beta_{[i]}i^a) \quad \text{for } i=k+1,\ldots,n. \end{aligned}$$

Therefore, the makespan of S is

$$C_{\max}(S) = C_{[n]}(S) \geq t_0 \prod_{j=1}^{k}(1+\alpha_{[j]}j^a) \prod_{j=k+1}^{n}(1+\alpha_{[j]}j^a)(1+\beta_{[i]}i^a). \quad (14)$$

Observe that the term $t_0 \prod_{j=1}^{k}(1+\alpha_{[j]}j^a)$ on the right-hand side of Eq. 14 is known and a lower bound for π can be obtained by minimizing the term $\prod_{j=k+1}^{n}(1+\alpha_{[j]}j^a)(1+\beta_{[i]}i^a)$. Lee [10] proves that the makespan is minimized by sequencing the jobs according to the shortest remaining deterioration rate (SRPT) rule. Thus, the minimal value of Eq. 14 can be obtained by sequencing the unscheduled jobs in non-decreasing order of the deterioration rates on M_1. Consequently, we obtain the second lower bound

$$LB_2 = t_0 \prod_{j=1}^{k}(1+\alpha_{[j]}j^a) \prod_{j=k+1}^{n}(1+\alpha_{(j)}j^a)\left(1+\min_{k+1\leq j\leq n}\beta_j n^a\right), \quad (15)$$

where $\alpha_{(k+1)} \leq \alpha_{(k+2)} \leq \cdots \leq \alpha_{(n)}$ is a non-decreasing order of the deterioration rates on M_1 for the remaining unscheduled jobs.

In order to make the lower bound tighter, we choose the maximum value between Eqs. (13) and (15) as the lower bound for π. That is,

$$LB = \max\{LB_1, LB_2\}.$$

4.3 The branch-and-bound algorithm

Our branch-and-bound algorithm uses the depth search strategy. We first perform the heuristic algorithm to obtain a sequence as the initial solution. This algorithm assigns jobs in a forward manner starting from the first position. In the searching tree, we choose a branch and systematically work down the tree until we either eliminate it on the basis of a lower bound or reach its final node, which is either used as a substitute for the initial solution or eliminated.

Branch-and-bound Algorithm

Step 1. Use Heuristic Algorithm 1 and Heuristic Algorithm 2 (if both solutions are different, then choose the best solution) to obtain an initial solution for the problem.

Step 2. Start the assignment of jobs at the beginning of a schedule and move forward one step at a time.

Step 3. In the kth level node, the first k positions are occupied by k specific jobs. Select one of the remaining $n-k$ jobs for the node at level $k+1$.

Step 4. First apply Theorems 1 and 2, then use Propositions 1.1-3.4 to eliminate the dominated partial sequences from the initial node and their descendants from the tree.

Step 5. Calculate the lower bound for the node. If the lower bound for an unfathomed partial schedule is larger than the initial solution, eliminate the node. Calculate the objective function value of the completed schedule, if it is less than the initial solution, replace it as the new solution; otherwise, eliminate it.

Step 6. Continue until all nodes have been explored, and the solution that ultimately remains is optimal.

5 Computational experiments

Computational experiments were conducted to evaluate the effectiveness of the branch-and-bound algorithm and the heuristic algorithms. The heuristic algorithm and the branch-and-bound algorithm were coded in VC++ 6.0 and the computational experiments were run on a Pentium 4 personal computer with a RAM size of 1 Gb. The job deterioration rates on machines M_1 and M_2 were generated from a uniform distribution over (0, 1).

In order to test the branch-and-bound algorithm, 12 different job sizes, n = 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30 and 32 were used. And 50 replications were randomly generated for each condition. A total of 600 problems were tested. For all the tests, we set $t_0 = 1$. In addition, the learning curves were taken to be 90%, 80%, and 70%, which yielded $a = -0.152$, -0.322, and -0.515, respectively, according to Biskup [2]. For the branch-and-bound algorithm, the average number of nodes, the maximum number of nodes, the average time, and the maximum time (in milliseconds) are reported. The contributions of the dominance properties and the lower bound are evaluated by algorithm efficiency, which is calculated in terms of the number of nodes explored compared with the total number of nodes. The percentage error of the solution produced by the heuristic algorithms are calculated as

$$(\mathrm{HA1} - V^*)/V^* \times 100\% \quad \text{and} \quad \left(\mathrm{HA2} - V^*\right)/V^* \times 100\%$$

where HA1 and HA2 are the makespan of the solution generated by the heuristic methods and V^* is the makespan of the optimal schedule. The running times for the heuristic algorithms are not given, since most problems are done in no time (a reported CPU time is zero) and the others are finished within a second. The contribution of the dominance properties and the lower bound is shown by the algorithm efficiency, which is calculated in terms of the number of nodes explored compared with the total number of nodes. The results are summarized in Table 1.

In Table 1, it is shown that the branch-and-bound algorithm can solve a problem of up to 32 jobs in a reasonable amount of time. However, the execution time and the number of nodes increase dramatically as the job size increases. The computational results also show that the proposed heuristics perform very well in terms of error percentages. The mean error percentage is less than 0.1% for all sizes of the problems. From Table 1, we also see that HA2 is more effective than HA1 when $a = -0.322$ and -0.515.

Table 1 Results for branch-and-bound algorithm and heuristic algorithms

n	a	Branch and bound algorithm				Error percentage			
		CPU time (ms)		Node number		HA1		HA2	
		Mean	Max	Mean	Max	Mean	Max	Mean	Max
	−0.152	2.4	16	48.5	514	0.005	0.035	0.016	0.064
10	−0.322	3.1	16	15.85	76	0.01	0.082	0.009	0.072
	−0.515	1.55	16	13.5	39	0.02	0.114	0.018	0.088
	−0.152	9.4	79	101.4	1538	0.004	0.054	0.026	0.404
12	−0.322	4.65	16	19.75	117	0.007	0.08	0.006	0.075
	−0.515	3.9	16	19.4	95	0.024	0.105	0.019	0.082
	−0.152	12.45	78	67.6	619	0.005	0.037	0.009	0.047
14	−0.322	7.85	31	29.95	181	0.018	0.117	0.013	0.046
	−0.515	10.95	47	63.85	534	0.026	0.121	0.015	0.068
	−0.152	9.4	31	14.05	57	0.011	0.062	0.015	0.087
16	−0.322	17.9	62	33.95	215	0.013	0.099	0.012	0.082
	−0.515	17.2	47	31.15	187	0.04	0.157	0.007	0.041
	−0.152	21.05	47	15.4	60	0.012	0.076	0.021	0.074
18	−0.322	29.65	157	55.25	600	0.039	0.264	0.015	0.087
	−0.515	17.2	62	25.7	178	0.032	0.098	0.013	0.067
	−0.152	71.3	500	113.65	1113	0.007	0.061	0.028	0.144
20	−0.322	81.15	531	150.65	1416	0.024	0.116	0.017	0.082
	−0.515	50.75	172	55.35	304	0.05	0.178	0.009	0.047
	−0.152	71.2	313	49.75	392	0.01	0.031	0.024	0.211
22	−0.322	63.35	188	41.85	161	0.049	0.172	0.006	0.031
	−0.515	80.25	359	69.85	504	0.046	0.277	0.007	0.074
	−0.152	82.75	219	48.85	222	0.008	0.054	0.017	0.078
24	−0.322	91.3	406	68.35	479	0.02	0.062	0.01	0.076
	−0.515	114.95	360	64.85	293	0.038	0.213	0.004	0.022
	−0.152	209.45	1281	139.8	1584	0.019	0.123	0.021	0.158
26	−0.322	451.5	5422	222.55	3170	0.044	0.145	0.017	0.121
	−0.515	217.15	875	94.4	530	0.049	0.172	0.012	0.068
	−0.152	185.9	609	62.8	308	0.011	0.1	0.032	0.146
28	−0.322	317.95	2734	148.05	1597	0.047	0.185	0.018	0.148
	−0.515	142.3	359	43.6	192	0.028	0.107	0.011	0.051
	−0.152	504.1	6906	324.05	5816	0.012	0.1	0.015	0.112
30	−0.322	231.3	610	64.1	242	0.045	0.164	0.013	0.076
	−0.515	832.75	4110	364.95	2600	0.064	0.394	0.01	0.037
	−0.152	487.45	2172	112.45	919	0.015	0.105	0.024	0.15
32	−0.322	946.1	6828	276.3	3002	0.041	0.24	0.027	0.204
	−0.515	317.1	766	64.5	227	0.068	0.221	0.022	0.122

6 Conclusions

We study in this paper a two-machine flow shop scheduling problem under the assumption that the processing times are not constant over time. We assume that processing time of a job is a function of its execution starting time and its position in a sequence. The computational complexity of this problem remains an open problem. Several dominance conditions and lower bounds for the makespan of the problem are developed and used in a proposed branch-and-bound algorithm to search for the optimal solutions. Two heuristic algorithms are also proposed, which are shown by computational experiments to be effective and efficient in obtaining near-optimal solutions. Extension to bi-criterion or other objective functions seems to be an interesting topic for future research.

Acknowledgments We are grateful to two anonymous referees for their helpful comments on an earlier version of this paper. We acknowledge The Hong Kong Polytechnic University for financial support under the project G-YJ35. This research was also supported in part by the National Natural Science Foundation of China under grant number 11001181.

References

1. Alidaee, B., Womer, N.K.: Scheduling with time dependent processing processing times: review and extensions. J. Oper. Res. Soc. **50**, 711–720 (1999)
2. Biskup, D.: Single-machine scheduling with learning considerations. Eur. J. Oper. Res. **115**, 173–178 (1999)
3. Biskup, D.: A state-of-the-art review on scheduling with learning effects. Eur. J. Oper. Res. **188**, 315–329 (2008)
4. Cheng, T.C.E., Ding, Q., Lin, BMT.: A concise survey of scheduling with time-dependent processing times. Eur. J. Oper. Res. **152**, 1–13 (2004)
5. Cheng, T.C.E., Wu, C.-C., Lee, W.C.: Some scheduling problems with sum-of-processing -times-based and job-position-based learning effects. Inf. Sci. **178**, 2476–2487 (2008)
6. Cheng, T.C.E., Wu, C.-C., Lee, W.C.: Some scheduling problems with deteriorating jobs and learning effects. Comput. Ind. Eng. **54**, 972–982 (2008)
7. Floudas, C.A., Pardalos, P.M.: Encyclopedia of Optimization. 2nd ed. Springer (2009)
8. Gawiejnowicz, S.: Time-dependent scheduling. Springer (2008)
9. Hadda, H.: A $(\frac{4}{3})$-approximation algorithm for a special case of the two machine flow shop problem with several availability constraints. Optim. Lett. **3**, 583–592 (2009)
10. Lee, W.-C.: A note on deteriorating jobs and learning in single-machine scheduling problems. Int. J. Bus. Econ. **3**, 83–89 (2004)
11. Lee, W.-C., Wu, C.-C.: Minimizing total completion time in a two-machine flowshop with a learning effect. Int. J. Prod. Econ. **88**, 85–93 (2004)
12. Lee, W-C., Wu, C-C.: A two-machine flowshop makespan scheduling problem with deteriorating jobs. Comput. Ind. Eng. **54**, 737–749 (2008)
13. Lee, W-C., Wu, C-C., Hsu, P-H.: A single-machine learning effect scheduling problem with release times. Omega Int. J. Manag. Sci. **38**, 3–11 (2010)
14. Lee, C.-Y., Yu, G.: Parallel-machine scheduling under potential disruption. Optim. Lett. **2**, 27–37 (2008)
15. Mosheiov, G.: Complexity analysis of job-shop scheduling with deteriorating jobs. Discret. Appl. Math. **117**, 195–209 (2002)
16. Nembhard, DA., Osothsilp, N.: Task complexity effects on between-individual learning/forgetting variability. Int. J. Ind. Ergonomics **29**, 297–306 (2002)
17. Ng, C.T., Wang, J.-B., Cheng, T.C.E., Liu, L.-L.: A branch-and-bound algorithm for solving a two-machine flow shop problem with deteriorating jobs. Comput. Oper. Res. **37**, 83–90 (2010)
18. Pardalos, P.M., Resende, M.G.C.: Handbook of Applied Optimization. Oxford Univ Press, USA (2002)
19. Pardalos, P.M.: Complexity in Numerical Optimization. World Scientific, USA (1993)

20. Setamaa-Karkkainen, A., Miettinen, K., Vuori, J.: Heuristic for a new multiobjective scheduling problem. Optim. Lett. **1**, 213–225 (2007)
21. Shiau, Y.-R., Lee, W.-C., Wu, C.-C., Chang, C.-M.: Two-machine flowshop scheduling to minimize mean flow time under simple linear deterioration. Int. J. Adv. Manuf. Technol. **34**, 774–782 (2007)
22. Toksar, M.D., Guner, E.: Minimizing the earliness/tardiness costs on parallel machine with learning effects and deteriorating jobs: a mixed nonlinear integer programming approach. Int. J. Adv. Manuf. Technol. **38**, 801–808 (2008)
23. Wang, C., Chu, C., Proth, J.M.: Efficient heuristic and optimal approaches for $n/2/F/\sum C_i$ scheduling problems. Int. J. Prod. Econ. **44**, 225–237 (1996)
24. Wang, J.-B., Liu, L.-L.: Two-machine flow shop problem with effects of deterioration and learning. Comput. Ind. Eng. **57**, 1114–1121 (2009)
25. Wang, J.-B., Wang, M.-Z.: Single-machine scheduling with nonlinear deterioration. Optim. Lett. doi:10.1007/s11590-010-0253-3
26. Wang, X.-Y., Wang, D., Yin, N.: A note on single-machine scheduling with nonlinear deterioration. Optim. Lett. doi:10.1007/s11590-011-0295-1
27. Wu, C.-C., Lee, W.C.: Single-machine scheduling problems with a learning effect. Appl. Math. Model. **32**, 1191–1197 (2008)
28. Wu, C.C., Lee, W.C.: Single-machine group scheduling problems with deteriorating setup times and job processing times. Int. J. Prod. Econ. **115**, 128–133 (2008)
29. Wu, C.-C., Lee, W.-C., Shiau, Y.-R.: Minimizing the total weighted completion time on a single machine under linear deterioration. Int. J. Adv. Manuf. Technol. **33**, 1237–1243 (2007)
30. Zhao, C.-L., Tang, H.-Y.: A note on two-machine no-wait flow shop scheduling with deteriorating jobs and machine availability constraints.. Optim. Lett. **5**, 183–190 (2011)