RAIRO-Oper. Res. 50 (2016) 211–219
DOI: 10.1051/ro/2015026 **RAIRO Operations Research**
www.rairo-ro.org

A BRANCH AND BOUND ALGORITHM TO MINIMIZE THE SINGLE MACHINE MAXIMUM TARDINESS PROBLEM UNDER EFFECTS OF LEARNING AND DETERIORATION WITH SETUP TIMES

M. Duran Toksari[1]

Abstract. This paper sheds light on minimizing the maximum tardiness with processing and setup times under both learning effect and deterioration. In this paper, all the jobs have processing and setup times under effects of learning and deterioration. By the effects of learning and deterioration, we mean that the processing time of a job is defined by an increasing function of its execution start time and position in the sequence. We provide a branch and bound algorithm to minimize the maximum tardiness under effects of learning and deterioration with setup times. Computational experiments show that the proposed algorithm can solve instances up to 800 jobs in reasonable time.

Mathematics Subject Classification. 90B35, 90B36.

Received August 5, 2014. Accepted June 12, 2015.

1. Introduction

In classical scheduling problems, the processing times aren't considered under some effects. However, in many real world scheduling environments such as steel production, iron ingots, fire-fighting *etc.*, both setup time of any family and processing time of any job increase or decrease over time. If a job is processed later, it consumes more time than the job when it is processed earlier. In this case, the job is under deterioration effect. Gupta and Gupta [5], and Browne and Yechiali [3] introduced independently deterioration effect on jobs in scheduling problems. Alidae and Womer [1] classified deterioration effect models into three different types: linear, piecewise linear and non-linear. In this paper, we consider linear effect for both job processing times and family setup times in single machine scheduling problems to minimize maximum tardiness by a branch and bound algorithm. We assume that the actual processing time of job j under linear deterioration effect is such that

$$\hat{p}_j = (p_j + (\alpha \times t_j)) \tag{1.1}$$

where p_j is the basic processing time of job j, α is common deterioration effect, and t_j is the starting time of job j.

On the other hand, in many realistic scheduling environments, workstations speed continuously up as a result of learning for repeating the same or similar activities. Thus, the processing time of a job is shorter if it is scheduled later in the sequence [15, 16]. In the literature, this phenomenon, which was first entitled by

Keywords. Maximum tardiness problem, learning effect, deterioration effects, setup times.

[1] Erciyes University, Engineering Faculty, Industrial Engineering Department, Kayseri, Turkey. dtoksari@erciyes.edu.tr

Mosheiov [14], is known as a "learning effect". We assume that the actual processing time of job j under learning effect is such that

$$\hat{p}_j = p_j \times r^a \tag{1.2}$$

where p_j is the basic processing time of job j, r is position into sequence of job j, and a is common learning effect.

In the scheduling literature, many researchers [8, 9, 11, 12, 17–27, 29, 30, 37] have simultaneously used deterioration effect and learning effect to optimize scheduling problems. Furthermore, some scheduling problems under learning effect and/or deterioration effect are extended to group scheduling problems [11,32,34–36]. In the some problems, there is no setup time between two consecutive jobs in the same group. However, each group has a group setup time to set up the tools, jigs and fixture on machines [10]. Some researchers [4,28,31,33] have worked on group scheduling problem with setup times under learning effect and/or deterioration effect. Wang *et al.* [28] mean that the group setup times is an increasing function of its starting time. Wu and Lee [31] considered which setup times are lengthened as jobs wait to be processed. Cheng *et al.* [4] minimized the maximum tardiness with deteriorating setup times. The problem under study in Xingong and Guangle [33] used setup times under learning effect for single machine group scheduling problems. In this paper, we used different setup times for each group.

The rest of the paper is organized as follows: in Section 2, we present the problem formulation. In Section 3, we show a branch and bound algorithm and a lower bound. In Section 4, we present the computational experiment. The conclusions of the research are summarized in Section 5.

2. Problem formulation

In the single machine environment, we have n jobs to be classified into m families. All jobs are available at time zero. Each family has sequence-independent setup time. In our problem, we define the actual processing time under learning and deterioration effects as follows:

$$p_{[r]} = \left(p_k + (\alpha \times t)\right)(r)^a \tag{2.1}$$

where p_k and $p_{[r]}$ are the basic processing time and the actual processing time of job k, which schedule in rth position, respectively. t is the starting time of the job, a is common learning effect for all jobs, and α is common deterioration effect for all jobs. Moreover, we assume that the actual setup time of a job from family i under learning and deterioration effects can be expressed as follows:

$$s_{[i]} = \left(s_i + (\theta \times t)\right)(R)^b \tag{2.2}$$

where s_i and $s_{[i]}$ are the basic setup time and the actual setup time of family i, respectively. R, t are the position and starting time of family i, respectively, b is common learning effect for all family, and θ is common deterioration effect for all jobs.

Cheng *et al.* [4] proposed a branch and bound algorithm to solve maximum tardiness with deteriorating jobs and setup times. There are two important differences between this paper and [4]. Firstly, the structure of the deterioration effect used in [4] is different from this paper. Cheng *et al.* [4] assume that the actual job processing time of job j is a simple linear function of its starting time t such that

$$p_j = \alpha_j \times t. \tag{2.3}$$

However, the deterioration effect for a job in this paper is common, and bases on both its starting time and simple processing time.

Secondly, the problem worked by Cheng *et al.* [4] is without learning effect. The processing time of a job under learning effect is shorter if it is scheduled later. On the other hand, if a job under deterioration is processed later, it consumes more time than the job when it is processed earlier. In this paper, we use simultaneously these reverse effects when Cheng *et al.* [4] consider only deterioration effect.

3. A branch and bound algorithm

Cheng *et al.* [4] show that single-machine scheduling problem with deterioration jobs and setup times for minimizing the maximum tardiness is NP-hard. Thus we propose a branch and bound algorithm to minimize the maximum tardiness with processing times and setup times under both learning effect and deterioration. We present first some dominance properties and a lower bound, and then we introduce the branch and bound algorithm in detail.

3.1. Dominance properties

In this section, we propose some rules to eliminate the dominated sequences.

Theorem 3.1. *If jobs j and k are into the family i, and the relations between their processing times and due dates are $p_j^i < p_k^i$ and $d_j^i \leqslant d_k^i$, respectively, then job j precedes job k in an optimal sequence.*

Proof. Let $S^i = (\pi^i, J_j^i, J_k^i, \hat{\pi}^i)$ show a sequence in family G_i where J_j and J_k are scheduled in the rth and $(r+1)$th positions, respectively. When schedule is $\hat{S}^i = (\pi^i, J_k^i, J_j^i, \hat{\pi}^i)$, J_j and J_k are scheduled in the rth and $(r+1)$th positions, respectively. The tardiness of the jobs in the partial sequence $(\pi^i, \hat{\pi}^i)$ is the same in both sequences since jobs J_j and J_k are from the family G_i. We use a similar approach as the one described in [17, 21, 22]. $T_{[r]}^i(S^i) = T_j^i(S^i)$ and $T_{[r]}^i(\hat{S}^i) = T_k^i(\hat{S}^i)$ are tardiness for the jobs in (r) th positions of both schedules, respectively. $T_{[r+1]}^i(S^i) = T_k^i(S^i)$ and $T_{[r+1]}^i(\hat{S}^i) = T_j^i(\hat{S}^i)$ are tardiness for the jobs in $(r+1)$th positions of both schedules, respectively. t is starting time of the job scheduled in position r.

$$T_{[r]}^i\left(S^i\right) = T_j^i\left(S^i\right) = \underbrace{\left(t + \left(p_j^i + \alpha.t\right)(r)^a\right)}_{C_{[r]}^i(S^i)} - d_j^i$$

$$T_{[r+1]}^i\left(S^i\right) = T_k^i\left(S^i\right) = \left(C_{[r]}^i\left(S^i\right) + \left(p_k^i + \alpha\left(C_{[r]}^i\left(S^i\right)\right)\right)(r+1)^a\right) - d_k^i$$

$$T_{[r]}^i\left(\hat{S}^i\right) = T_k^i\left(\hat{S}^i\right) = \underbrace{\left(t + \left(p_k^i + \alpha.t\right)(r)^a\right)}_{C_{[r]}^i(\hat{S}^i)} - d_k^i$$

$$T_{[r+1]}^i\left(\hat{S}^i\right) = T_j^i\left(\hat{S}^i\right) = \left(C_{[r]}^i\left(\hat{S}^i\right) + \left(p_j^i + \alpha\left(C_{[r]}^i\left(\hat{S}^i\right)\right)\right)(r+1)^a\right) - d_j^i.$$

If $d_j^i \leqslant d_k^i$, then we obtain $T_k^i(\hat{S}^i) < T_j^i(\hat{S}^i)$. That is, $T_j^i(\hat{S}^i) = \max(T_k^i(\hat{S}^i), T_j^i(\hat{S}^i))$ if $d_j^i \leqslant d_k^i$.

If $p_j^i < p_k^i$, then $C_k^i(S^i) < C_j^i(\hat{S}^i)$. Thus, if $d_j^i \leqslant d_k^i$ and $p_j^i < p_k^i$, then $T_k^i(S^i) < T_j^i(\hat{S}^i)$ and $T_j^i(S^i) < T_j^i(\hat{S}^i)$. Therefore, $\max\{T_{[r]}^i(S^i), T_{[r+1]}^i(S^i)\} \leqslant \max\{T_{[r]}^i(\hat{S}^i), T_{[r+1]}^i(\hat{S}^i)\}$ while $p_j^i < p_k^i$ and $d_j^i \leqslant d_k^i$.

Thus, $S^i = (\pi^i, J_j^i, J_k^i, \hat{\pi}^i)$ dominates $\hat{S}^i = (\pi^i, J_k^i, J_j^i, \hat{\pi}^i)$ and the proof is completed. □

We present three properties to use in the branch and bound algorithm, followed by a lower bound. We have used for all properties that $S^i = (\pi, J_j, J_k, \pi')$ and $\hat{S}^i = (\pi, J_k, J_j, \pi')$ are two sequences. For all properties, we assume $p_j^i < p_k^i$ and $d_j^i \leqslant d_k^i$. t is the starting time of the job scheduled in position r.

At this point, we introduce a proposition to determine the feasibility of a partial schedule to work better through the search process. Assume that (π, π^c) is sequence of jobs where π and π^c are the scheduled part and unscheduled part, respectively. $S^* = (\pi^*, \pi)$ be a sequence in which the unscheduled jobs (π^c) are arranged as follows: jobs, which are in the same family with the first job of π, are scheduled last, and if they are more than one, they are scheduled in the earliest due date (EDD) order. Other jobs are scheduled in the EDD order, if they are in the same family. Families are scheduled in the EDD order of the maximum due dates of the families. A similar approach was used in [4].

Theorem 3.2. *If $\max_{j\in\pi} T_j(S^*) \geqslant \max_{j\in\pi^*} T_j(S^*)$ then sequence (π^*, π) dominates sequences of type (π, π^c)* [4].

Proof. We know that S is a sequence of the type (π, π^c). We assume that $\max_{j\in\pi}\{T_j(S^*)\} \geqslant \max_{j\in\pi^*}\{T_j(S^*)\}$ then

$$\begin{aligned}\max_{1<j\leqslant n}\{T_j(S)\} &= \max\left\{\max_{j\in\pi}\{T_j(S)\}, \max_{j\in\pi^c}\{T_j(S)\}\right\} \geqslant \max_{j\in\pi}\{T_j(S)\} \\ &\geqslant \max\left\{\max_{j\in\pi}\{T_j(S^*)\}, \max_{j\in\pi*}\{T_j(S^*)\}\right\} = \max_{j\in\pi}\{T_j(S^*)\}.\end{aligned}$$

Thus, (π^*, π) is optimal among sequences of type (π, π^c) [4]. □

Property 1. *If jobs (j,k) are into family i, $A = t+\alpha tr^a+\alpha t(r+1)^a+\alpha^2 tr^a(r+1)^a$ and $d^i_j < A+(p^i_j(r+1)^a)+((p^i_k r^a)(1+\alpha(r+1)^a))$, then S^i dominates $\hat{S}^i$.*

Proof. $C^i_{[r]}(S^i) = C^i_j(S^i)$ and $C^i_{[r]}(\hat{S}^i) = C^i_k(\hat{S}^i)$ are completion times for the jobs in (r)th positions of schedules S^i and $\hat{S}^i$, respectively. $C^i_{[r+1]}(S^i) = C^i_k(S^i)$ and $C^i_{[r+1]}(\hat{S}^i) = C^i_j(\hat{S}^i)$ are completion times for the jobs in $(r+1)$th positions of schedules S^i and $\hat{S}^i$, respectively.

$$\begin{aligned}C^i_{[r]}\left(S^i\right) &= C^i_j\left(S^i\right) = t + \left(p^i_j + \alpha t\right)(r)^a \\ C^i_{[r+1]}\left(S^i\right) &= C^i_k\left(S^i\right) = C^i_{[r]}\left(S^i\right) + \left(p^i_k + \alpha\left(C^i_{[r]}\left(S^i\right)\right)\right)(r+1)^a \\ C^i_{[r]}\left(\hat{S}^i\right) &= C^i_k\left(\hat{S}^i\right) = t + \left(p^i_k + \alpha t\right)(r)^a \\ C^i_{[r+1]}\left(\hat{S}^i\right) &= C^i_j\left(\hat{S}^i\right) = C^i_{[r]}\left(\hat{S}^i\right) + \left(p^i_j + \alpha\left(C^i_{[r]}\left(\hat{S}^i\right)\right)\right)(r+1)^a \\ &= A + \left(p^i_j(r+1)^a\right) + \left(\left(p^i_k r^a\right)\left(1+\alpha(r+1)^a\right)\right)\end{aligned}$$

then we have $\max\{T^i_k(S^i), T^i_j(S^i)\} \leqslant T^i_j(\hat{S}^i)$ where

$$d^i_j < A + \left(p^i_j(r+1)^a\right) + \left(\left(p^i_k r^a\right)\left(1+\alpha(r+1)^a\right)\right),\ p^i_j < p^i_k$$

and $d^i_j \leqslant d^i_k$ and the proof is completed. □

Recall that we obtained the setup time of family i as $s_{[i]} = (s_i + \theta t)(R)^b$ before Property 2 and Property 2 is presented.

Property 2. *If the last job in π is not into family i, $A' = (t+s_{[i]})(1+\alpha+\alpha 2^a+\alpha^2 2^a)$ and $d^i_j < A' + 2^a(p^i_j + \alpha p^i_k)$, then S^i dominates $\hat{S}^i$.*

Proof. If the last job in π is not family i, then the completion times of jobs (j,k) in S^i and $\hat{S}^i$ are

$$\begin{aligned}C^i_{[1]}\left(S^i\right) &= C^i_j\left(S^i\right) = t + s_{[i]} + \left(p^i_j + \alpha\left(t+s_{[i]}\right)\right)(1)^a \\ C^i_{[2]}\left(S^i\right) &= C^i_k\left(S^i\right) = C^i_{[1]}\left(S^i\right) + \left(p^i_k + \alpha\left(C^i_{[1]}\left(S^i\right)\right)\right)(2)^a \\ C^i_{[1]}\left(\hat{S}^i\right) &= C^i_k\left(\hat{S}^i\right) = t + s_{[i]} + \left(p^i_k + \alpha\left(t+s_{[i]}\right)\right)(1)^a \\ C^i_{[2]}\left(\hat{S}^i\right) &= C^i_j\left(\hat{S}^i\right) = C^i_{[1]}\left(\hat{S}^i\right) + \left(p^i_j + \alpha\left(C^i_{[1]}\left(\hat{S}^i\right)\right)\right)(2)^a = A' + 2^a\left(p^i_j + \alpha p^i_k\right)\end{aligned}$$

then we have $\max\{T^i_k(S^i), T^i_j(S^i)\} \leqslant T^i_j(\hat{S}^i)$ where $d^i_j < A' + 2^a(p^i_j + \alpha p^i_k)$, $p^i_j < p^i_k$ and $d^i_j \leqslant d^i_k$ and the proof is completed. □

Property 3. *If there is no job in π that is into family i,*

$$A'' = \left(\left(t + s_{[i]}\right)\left(1 + \alpha r^a + \alpha\left(r+1\right)^a\right)\right) + \left(\left(r^a\left(r+1\right)^a\right)\left(\alpha^2 t s_{[i]} + \alpha^2 s_{[i]}^2\right)\right)$$

and

$$d_j^i < A'' + \left(p_j^i\left(r+1\right)^a\right) + \left(\left(p_k^i r^a\right)\left(1 + \alpha\left(r+1\right)^a\right)\right),$$

then S^i dominates $\hat{S}^i$.

Proof. If the last job in π is not family i, then the completion times of jobs (j,k) in S^i and $\hat{S}^i$.

$$\begin{aligned}
C_{[r]}^i\left(S^i\right) &= C_j^i\left(S^i\right) = t + s_{[i]} + \left(p_j^i + \alpha\left(t + s_{[i]}\right)\right)(r)^a \\
C_{[r+1]}^i\left(S^i\right) &= C_k^i\left(S^i\right) = C_{[r]}^i\left(S^i\right) + \left(p_k^i + \alpha\left(C_{[r]}^i\left(S^i\right)\right)\right)(r+1)^a \\
C_{[r]}^i\left(\hat{S}^i\right) &= C_k^i\left(\hat{S}^i\right) = t + s_{[i]} + \left(p_k^i + \alpha\left(t + s_{[i]}\right)\right)(r)^a \\
C_{[r+1]}^i\left(\hat{S}^i\right) &= C_j^i\left(\hat{S}^i\right) = C_{[r]}^i\left(\hat{S}^i\right) + \left(p_j^i + \alpha\left(C_{[r]}^i\left(\hat{S}^i\right)\right)\right)(r+1)^a \\
&= A'' + \left(p_j^i\left(r+1\right)^a\right) + \left(\left(p_k^i r^a\right)\left(1 + \alpha\left(r+1\right)^a\right)\right) \\
d_j^i &< A'' + \left(p_j^i\left(r+1\right)^a\right) + \left(\left(p_k^i r^a\right)\left(1 + \alpha\left(r+1\right)^a\right)\right)
\end{aligned}$$

then we have $\max\{T_k^i(S^i), T_j^i(S^i)\} \leqslant T_j^i(\hat{S}^i)$ where

$$d_j^i < A'' + \left(p_j^i\left(r+1\right)^a\right) + \left(\left(p_k^i r^a\right)\left(1 + \alpha\left(r+1\right)^a\right)\right),\ p_j^i < p_k^i$$

and $d_j^i \leqslant d_k^i$ and the proof is completed. □

3.2. A lower bound

We developed, by inspiring from [4], a lower bound for minimizing the maximum tardiness problem with processing times and setup times under both learning effect and deterioration.

Let S^* be a partial schedule (PS) in which the order of the last (n-k) jobs is known and in which the first k jobs are unscheduled. Let $p_1 \leqslant p_2 \leqslant \ldots \leqslant p_k$ be the basic processing times of the unscheduled jobs, in non-decreasing order and let $d_1 \leqslant d_2 \leqslant \ldots \leqslant d_k$ their due dates, also in non-decreasing order. Note that p_1 and d_1 may not belong to the same job. Further, let $n_1 \leqslant n_2 \leqslant \ldots \leqslant n_m$ be the sizes of the families associated with the unscheduled jobs, and let $s_1 \leqslant s_2 \leqslant \ldots \leqslant s_m$ be their setup times, both of them in non-decreasing order Note that n_1 and s_1 may not belong to the same family.

The actual completion time of the first job is $C_{[1]}^i = t + s_{[i]} + (p_1^i + \alpha(t_1^i + s_{[i]}))(1)^a$ where setup time of family i is $s_{[i]} = (s_i + \theta t)(R)^b$. Similarly, the completion time of the rth job (if job k is scheduled in position r) is

$$C_{[r]} = C_q = \underbrace{t_0 + \sum_{u=1}^{i-1}\left(s_{[u]} + \sum_{v=1}^{n_u} p_{[v]}^u\right) + s_{[i]} + \sum_{z=1}^{r-1} p_{[z]}^i}_{t_q^i} + \left(p_q^i + \alpha\left(t_q^i + s_{[i]}\right)\right)(r)^a \qquad 1 \leqslant r \leqslant k.$$

We can derive a lower bound on the completion time of the kth job by assuming that jobs into the first family with smallest basic processing times form a family with setup times $s_{[1]}$, the next family with the second smallest basic processing times form a family with setup times $s_{[2]}$, and so on. Thus, we have

$$C_{[r]} \geqslant C_{[r]}^* = t_0 + \sum_{u=1}^{i-1}\left(s_{[u]} + \sum_{v=1}^{l_r} p_{[v]}^u\right) + s_{[i]} + \sum_{v=1}^{l_r} p_{[v]}^i \qquad 1 \leqslant r \leqslant k$$

where l_r is the smallest number such that $r \leqslant n_{[1]} + n_{[2]} + \ldots + n_{[l_r]}$. We can compute a lower bound on the completion times of the scheduled jobs accordingly. That is,

$$C_{[r]} \geqslant C^*_{[r]} = C_{[k]} + \sum_{u=i+1}^{m-1} \left(s_{[u]} + \sum_{v=1}^{n_u} p^u_{[v]} \right) + s_{[m]} + \sum_{z=1}^{r-1} p^m_{[z]} + \left(p^i_q + \alpha \left(t^i_q + s_{[i]} \right) \right) (r)^a \qquad k+1 \leqslant r \leqslant n.$$

Thus, we have

$$\begin{aligned} T_{\max}(PS) &= \max \left\{ \max_{1 \leqslant r \leqslant k} \left\{ C_{[r]}(PS) - d_{[j]} \right\}, \max_{k+1 \leqslant r \leqslant n} \left\{ C_{[r]}(PS) - d_{[j]} \right\}, 0 \right\} \\ &\geqslant \max \left\{ \max_{1 \leqslant r \leqslant k} \left\{ C^*_{[r]} - d_{[j]} \right\}, \max_{k+1 \leqslant r \leqslant n} \left\{ C^*_{[r]} - d_{[j]} \right\}, 0 \right\}. \end{aligned}$$

This inequality holds because $C^*_{[r]}$ is a lower bound on the actual completion times. Therefore, a lower bound on the maximum tardiness of the PS is

$$LB = \max \left\{ \max_{1 \leqslant r \leqslant k} \left\{ C^*_{[r]} - d_{[j]} \right\}, \max_{k+1 \leqslant r \leqslant n} \left\{ C^*_{[r]} - d_{[j]} \right\}, 0 \right\}.$$

3.3. The branch and bound algorithm

We used the depth-first search in the branching procedure. A similar approach was used in [4,13,16]. According to this approach, the algorithm only needs to store the lower bounds for at most $n-1$ active nodes through the branch procedure. The algorithm assigns jobs in a forward manner starting from the first position. The steps of the branch-and-bound algorithm are described as follows.

Step 1. Arrange the initial solution $S = (-, -, \ldots -)$ with maximum tardiness ∞.
Step 2. Apply Theorem 3.1 and Properties 1–3 to eliminate the dominated partial sequences.
Step 3. For non-dominated nodes, apply Theorem 3.2 to find the order for unscheduled jobs.
Step 4. Find the lower bound of the maximum tardiness for the unscheduled partial sequences or the maximum tardiness for the completed sequences. If the lower bound for the unscheduled partial sequence is greater than or equal the initial solution, eliminate the node and all nodes beyond it in the branch. If the value of the completed sequence is less than the initial solution, replace it as the new solution. Otherwise, eliminate it.
Step 5. Repeat Steps 2–4 until all trees have been completed.

4. Computational experiments

We conducted computational experiments in order to evaluate the performance of the optimal branch and bound algorithm. The proposed algorithms were coded in Visual Studio C#, and run on a PC with 2.33 GHz CPU and 3 GB RAM. We used three different numbers of jobs (n = 200, 500 and 800). The basic processing times were generated from a uniform distribution over the integers between 10 and 70 in every case, while the due dates were generated from a uniform distribution over the integers on (0, $15n\lambda$) where n is the number of jobs. Two different sets of problem instances were generated by giving λ the values 0.06 and 0.08. Moreover, the basic setup times were generated from a uniform distribution over the integers between 2 and 15 in every case. We assumed that the learning rate is 80% and 70%.

In this case, the learning effect is found using $a\,(=\log_2 e \leqslant 0)$ when e is the learning rate. By the literature, we know that different products have different learning curves, and that the rate of learning varies depending upon the potential of the process and product [6]. So, the learning rates vary between 70% and 90% for different manufacturing industries in the US between 1920 and 1988. For examples; 79% and 80% were in the steel industry and aircraft assembly, respectively [16]. On the other hand, some researchers [2, 7, 16] used $(0, 1]$ for determination of deterioration rate, so we assumed deterioration rates of the job processing times $\alpha = 0.1$ and

TABLE 1. The performance of proposed branch and bound algorithm.

n	e	α	λ	Branch and bound algorithm					
				CPU time (s)			Number of nodes		
				Mean	SD	Max	Mean	SD	Max
200	80%	0.1	0.06	0.0565	0.0408	0.0351	353.6900	263.0323	1194
			0.08	0.0781	0.0664	0.0610	556.1400	401.7436	1218
		0.2	0.06	0.0304	0.0312	0.0283	246.1100	173.1940	781
			0.08	0.1285	2.1154	14.6257	1351.1900	7714.6406	45 893
	70%	0.1	0.06	0.0667	0.0591	0.0490	472.8600	241.9080	1547
			0.08	0.0712	0.0704	0.0657	566.6800	400.1145	1805
		0.2	0.06	0.0518	0.0488	0.0351	367.0700	165.4402	1201
			0.08	0.1255	9.2516	15.2657	3598.6100	8903.5885	83 672
500	80%	0.1	0.06	0.4512	0.3858	1.4576	654.5000	1200.1674	8736
			0.08	0.9658	4.5268	14.6215	5427.6600	14 788.8537	75 327
		0.2	0.06	0.7091	0.6654	3.0025	858.9600	1909.0245	7176
			0.08	7.3652	18.8873	87.6302	5051.3500	19 256.3935	97 034
	70%	0.1	0.06	0.8856	0.7451	2.9573	1057.9800	2386.8947	9701
			0.08	1.0291	22.0276	114.2592	7283.4700	23 721.2232	102 694
		0.2	0.06	0.9912	0.8886	4.5614	1020.8900	2603.4232	7571
			0.08	8.2650	72.9154	433.8504	9381.7940	28 608.4713	159 697
800	80%	0.1	0.06	1.5874	1.6238	9.9566	1190.5600	1423.0420	8796
			0.08	22.1547	101.7452	994.2318	7125.5000	12 488.3670	35 056
		0.2	0.06	2.6547	2.0581	18.6254	1806.4500	1406.5084	11 374
			0.08	39.2514	185.6183	2564.8400	8071.8000	60 732.0770	20 883
	70%	0.1	0.06	5.2194	5.2773	27.9932	1999.0800	1859.5783	9553
			0.08	18.8957	77.4599	924.6329	11 062.5560	45 425.8710	266 734
		0.2	0.06	8.6581	7.9228	28.6615	2121.1300	1652.0618	6536
			0.08	74.6215	254.2165	3324.5641	14 435.0200	50 411.9209	273 445

0.2, and we generated the deterioration rates of the family setup times (θ) from uniform distributions over 0 and 1. For each condition, 100 instances were randomly generated, and the results are presented in Table 1.

Table 1 shows that if the value of λ becomes large, the range of the job release times is large, thus Theorem 3.2 is useful in that case. This explains the decreasing trend of the number of nodes as the value of λ becomes large. The mean error percentages decreased to zero as the value of λ became large. This explains the increment of the number of nodes for small values of λ. Moreover, this is due to the fact that Theorem 3.2 is less potent when due dates are more variable and small, as explained earlier.

5. Conclusions

In this paper, we consider single machine scheduling problem with processing times and setup times under both learning effect and deterioration to minimize the maximum tardiness. Cheng *et al.* [4] show that single-machine scheduling problem with deterioration jobs and setup times for minimizing the maximum tardiness is NP-hard. Therefore, the problem with both learning effect and deterioration is NP-hard, indicating that finding an optimal solution is difficult. In this paper, we develop a branch and bound algorithm incorporating several dominances and a lower bound to solve the problem, which seems to work well for reasonably sized problems. The branch-and-bound algorithm performs well in terms of the number of nodes and the CPU time when the number of jobs is less than or equal to 800. Different performance measures such as other due date related objective functions can also be considered under these effects and batch setup times.

References

[1] B. Alidaee and N.K. Womer, Scheduling with time dependent processing times: Review and extensions. *J. Oper. Res. Soc.* **50** (1999) 711–720.

[2] A. Bachman, T.C.E. Cheng, A. Janiak and C.T. Ng, Scheduling start time dependent jobs to minimize the weighted total completion time. *J. Oper. Res. Soc.* **53** (2002) 668–693.

[3] S. Browne and U. Yechiali, Scheduling deteriorating jobs on a single processor. *Oper. Res.* **38** (1990) 495–498.

[4] T.C.E. Cheng, C.J. Hsu, Y.C. Huang and W.C. Lee, Single-machine scheduling with deteriorating jobs and setup times to minimize the maximum tardiness. *Comput. Oper. Res.* **38** (2011) 1760–1765.

[5] J.N.D. Gupta and S.K. Gupta, Single facility scheduling with nonlinear processing times. *Comput. Ind. Eng.* **14** (1988) 387–393.

[6] J. Heizer and B. Render, Operations Management, 6th edition. Prentice-Hall (2001).

[7] Y.S. Hsu and B.M.T. Lin, Minimization of maximum lateness under linear deterioration. *Omega: Int. J. Manag. Sci.* **31** (2003) 459–469.

[8] X. Huang and M.-Z. Wang, Parallel identical machines scheduling with deteriorating jobs and total absolute differences penalties. *Appl. Math. Model.* **35** (2011) 1349–1353.

[9] X. Huang, J.-B. Wang, L.-Y. Wang, W.-J. Gao and X.-R. Wang, Single machine scheduling with time-dependent deterioration and exponential learning effect. *Comput. Ind. Eng.* **58** (2010) 58–63.

[10] W.H. Kuo and D.L. Yang, Single machine group scheduling with a time-dependent learning effect. *Comput. Oper. Res.* **33** (2006) 2099–2112.

[11] W.-C. Lee and C.-C. Wu, A note on single-machine group scheduling problems with position-based learning effect. *Appl. Math. Model.* **33** (2009) 2159–2163.

[12] W.-C. Lee and C.-C. Wu, Some single-machine and m-machine flowshop scheduling problems with learning consideration. *Inf. Sci.* **179** (2009) 3885–3892.

[13] W.C. Lee, C.C. Wu and P.H. Hsu, A single-machine learning effect scheduling problem with release times. *Omega: Int. J. Manag. Sci.* **38** (2010) 3–11.

[14] G. Mosheiov, Scheduling problems with a learning effect. *Eur. J. Oper. Res.* **132** (2001) 687–693.

[15] M.D. Toksarı and E. Guner, 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** (2008) 801–808.

[16] M.D. Toksarı, A branch and bound algorithm for minimizing makespan on a single machine with unequal release times under learning effect and deteriorating jobs. *Comput. Oper. Res.* **38** (2011) 1361–1365.

[17] M.D. Toksarı and E. Güner, Parallel machine earliness/tardiness scheduling problem under the effects of position based learning and linear/nonlinear deterioration. *Comput. Oper. Res.* **36** (2009) 2394–2417.

[18] M.D. Toksarı and E. Güner, Scheduling problems with the nonlinear effects of learning and deterioration. *Int. J. Adv. Manuf. Technol.* **45** (2009) 801–807.

[19] M.D. Toksari, D. Oron and E. Güner, Single machine scheduling problems under the effects of nonlinear deterioration and time-dependent learning. *Math. Comput. Model.* **50** (2009) 401–406.

[20] M.D. Toksari, D. Oron and E. Güner, Some Scheduling Problems with Past Sequence Dependent Setup Times Under the Effects of Nonlinear Deterioration and Time-Dependent Learning. *RAIRO: RO* **44** (2010) 107–118.

[21] M.D. Toksarı and E. Güner, The common due date Early/tardy scheduling problem on a parallel machine under the effects of time dependent learning and linear/ nonlinear deterioration. *Expert Systems with Applications* **37** (2010) 92–112.

[22] M.D. Toksarı and E. Güner, Parallel machine scheduling problem to minimize the earliness/tardiness costs with learning effect and deteriorating jobs. *J. Intelligent Manuf.* **21** (2010) 843–851.

[23] B. Wang, C.T. Ng and T.C.E. Cheng, Single-machine scheduling with deteriorating jobs under a series-parallel graph constraint. *Comput. Oper. Res.* **35** (2008) 2684–2693.

[24] J.-B. Wang, L. Lin and F. Shan, Single machine group scheduling problems with deteriorating jobs. *Int. J. Adv. Manuf. Technol.* **39** (2008) 808–812.

[25] J.-B. Wang, Single machine scheduling with a time-dependent learning effect and deteriorating jobs. *J. Oper. Res. Soc.* **60** (2009) 583–586.

[26] J.-B. Wang, W.-J. Gao, L.-Y. Wang and D. Wang, Single machine group scheduling with general linear deterioration to minimize the makespan. *Int. J. Adv. Manuf. Technol.* **43** (2009) 146–150.

[27] J.-B. Wang, X. Huang, X.Y. Wang, N. Yin and L.Y. Wang, Learning effect and deteriorating jobs in the single machine scheduling problems. *Appl. Math. Model.* **33** (2009) 3848–3853.

[28] J.B. Wang, L. Lin and F. Shan, Single-machine group scheduling problems with deteriorating jobs. *Int. J. Adv. Manuf. Technol.* **39** (2008) 7–8.

[29] C.-C. Wu and W.-C. Lee, Single-machine group-scheduling problems with deteriorating setup times and job-processing times. *Int. J. Prod. Econ.* **115** (2008) 128–133.

[30] C.-C. Wu, Y.-R. Shiau and W.-C. Lee, Single-machine group scheduling problems with deterioration consideration. *Comput. Oper. Res.* **35** (2008) 1652–1659.

[31] C.C. Wu and W.C. Lee, Single-machine group scheduling problems with deteriorating setup times and job processing times. *Int. J. Prod. Econ.* **115** (2008) 128–133.

[32] C.C. Wu, Y.R. Shiau and W.C. Lee, Single-machine group scheduling problems with deterioration consideration. *Comput. Oper. Res.* **35** (2008) 1652–1659.

[33] Z. Xingong and Y. Guangle, Single machine scheduling problems with deteriorated and learning effect. *Appl. Math. Comput.* **216** (2010) 1259–1266.

[34] S.-J. Yang, Single-machine scheduling problems with both start-time dependent learning and position dependent aging effects under deteriorating maintenance consideration. *Appl. Math. Comput.* **217** (2011) 3321–3329.

[35] S.-H. Yang and J.-B. Wang, Minimizing total weighted completion time in a two-machine flow shop scheduling under simple linear deterioration. *Appl. Math. Comput.* **217** (2011) 4819–4826.

[36] W.H. Yang and S. Chand, Learning and forgetting effects on a group scheduling problem. *Eur. J. Oper. Res.* **187** (2008) 1033–1044.

[37] V.C.Y. Zhu, L. Sun, L. Sun and X. Li, Single machine scheduling time-dependent jobs with resource-dependent ready times. *Comput. Ind. Eng.* **58** (2010) 84–87.