Research Article

Scheduling with Learning Effects and/or Time-Dependent Processing Times to Minimize the Weighted Number of Tardy Jobs on a Single Machine

Jianbo Qian and George Steiner

Operations Management Area, DeGroote School of Business, McMaster University, Hamilton, ON, Canada L8S 4L8

Correspondence should be addressed to George Steiner; george.steiner4@gmail.com

Received 2 August 2013; Revised 23 October 2013; Accepted 23 October 2013

Academic Editor: Chin-Chia Wu

Copyright © 2013 J. Qian and G. Steiner. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

We consider single machine scheduling problems with learning/deterioration effects and time-dependent processing times, with due date assignment consideration, and our objective is to minimize the weighted number of tardy jobs. By reducing all versions of the problem to an assignment problem, we solve them in $O(n^4)$ time. For some important special cases, the time complexity can be improved to be $O(n^2)$ using dynamic programming techniques.

1. Introduction

Traditionally, the processing times of jobs in scheduling problems were fixed, regardless of the jobs' positions in the schedule or their starting times. In recent years, however, more and more researchers are investigating scheduling problems with variable processing times of two types: learning/deterioration effects and time-dependent processing times.

Learning Effect. As early as in the 1930s, Wright [1] noticed that in the aircraft industry the working costs per unit decreased with an increasing production output. He formulated the so-called 80% hypothesis, stating that with every redoubling of output the unit processing time decreases by 20%. This learning effect has a large impact on mass production. For example, it was reported that, in the semiconductor industry, the efficiency gains cause the price to drop by 10–30% [2]. Recently, new proof of the impact of learning effects from a new car-assembly plant was presented in [3].

Learning effects are important for production problems involving significant level of human activities, such as machine setup, machine cleaning, machine maintenance, machine failure removal, machine operating and controlling, machine data reading/understanding/interpretation, and all kinds of handwork. They are especially important when the

production environment changes. Such changes may include, among others, investments in new machines or replacement of equipment, new workers, workflow changes, and the acceptance of new jobs. Even small changes to the production environment like a software update, a new design of the format of important documents, and a new organization of the spare parts depot cause learning effects. Every time a worker needs to get used to a new circumstance, a learning experience will occur. Since there are usually high levels of such human involvement in scheduling environments which change frequently, and these activities are subject to learning, it is important to consider learning effects on scheduling problems [4].

Biskup [5] proposed the following model:

$$p_{[j]} = a_{[j]} j^c, \quad (1)$$

where $p_{[j]}$ and $a_{[j]}$, respectively, represent the actual processing time and “normal” processing time (i.e., without learning effect) of the job at the j th position in the scheduling sequence and $c \leq 0$ represents the rate of learning. It is easy to verify that if $c = \log_2 0.8 = -0.322$, it corresponds to the aforementioned 80% hypothesis.

If $c > 0$, then the model captures the effect of process deterioration [6]. Many researchers used an equivalent model $p_{ir} = p_i r^c$, where r represents the position of job i in

the schedule. Mosheiov [7] investigated scheduling problems with learning effects described by (1) on parallel machines and Mosheiov and Sidney [8] studied more general learning functions. Scheduling on parallel machines with a more general form of learning effect was studied in [9]. A comprehensive survey about the learning effect in scheduling problems can be found in [10]. Further reviews of scheduling models with positional effects and learning were presented in [11–13].

Time-Dependent Processing Time. In some other practical situations, the actual processing time of a job depends on its starting time. For example, Gupta et al. [14] modeled a multiple loan repayment problem in financial management as a scheduling problem with time-dependent processing times. In steel production, steel ingots are to be heated to the required temperature before rolling can begin. The time taken to heat depends on the size and current temperature of the ingot, which further depends on the amount of time the ingot has been waiting to be processed in. Another example is in forest-fire fighting, where the time required to control a fire is longer if the start of the fire fighting effort is delayed [15]. Some other empirical applications exist in hospital emergency room scheduling and crime scene response scheduling for police [16, 17].

The concept of time-dependent processing times was introduced by Browne and Yechiali [18]. One of the most commonly used models is

$$p_{[j]} = a_{[j]} + bS_{[j]}, \quad (2)$$

where $a_{[j]}$ also denotes the “normal” processing time, b is a constant, and $S_{[j]}$ denotes the starting time of the job in the j th position. If $b > 0$, then it reflects deteriorating processing time. Mosheiov presented a V-shape policy and a $\wedge$ -shape policy, respectively, in [17, 19] for scheduling deteriorating jobs. Some researchers used position-dependent $b_{[j]}$ [20]. However, the problem usually becomes significantly harder in most of those cases. Gawiejnowicz gives a comprehensive exposition of time-dependent models [21].

The two types of variable processing times defined above were usually considered separately. Lee [22] was the first to consider them simultaneously. Wang [23] proposed the following combined model:

$$p_{[j]} = (a_{[j]} + bS_{[j]})j^c, \quad (3)$$

where $p_{[j]}$, $S_{[j]}$, b , and $c \leq 0$ all have the same meaning as before. This was followed by studies by Wang [24], Wang and Cheng [25], Wang et al. [26], Wang and Cheng [27], and Koulamas et al. [28].

This model reflects, for example, the following practical scenario: an operator obtains additional skills by learning from experience; at the same time, the machine that he/she operates may be subjected to wear and tear, that is, deteriorates with time [13]. A recent review by Janiak et al. focused on a variety of models with these types of combined effects [12].

We can see that each of the above discussed cases of variable processing times is a special case of this general

model. If $b = 0$, we have the learning effect problem; if $c = 0$, we have the time-dependent processing time problem.

The general problem we study in this paper may be stated as follows: n independent, nonpreemptive jobs, $J = \{1, 2, \dots, n\}$, are available for processing at time zero and are to be processed on a single machine. A schedule is defined by a job sequence $\pi = ([1], [2], \dots, [n])$, where $[j]$ represents the job that is in the j th position in π for $j = 1, 2, \dots, n$. The processing time of the job at position j is defined by (3). Our objective is to determine a schedule which minimizes a general unified cost function that is the sum of scheduling costs, expressed by using positional penalties. This unified cost function can be formulated as follows:

$$g(\pi) = \sum_{j=1}^n \xi_j \eta_{[j]} + \sum_{j=l+1}^n \psi_{[j]}, \quad (4)$$

where ξ_j is a *positional, job-independent penalty* for any job scheduled in the j th position for $j = 1, \dots, n$, $\eta_{[j]}$ represents the normal processing time of the job in position j , l is a variable index to be determined, and $\psi_{[j]}$ is the scheduling penalty, for example, for tardiness, incurred by jobs scheduled in the last $n - l$ positions. Our model is adapted from [29], where Leyvand et al. proposed a unified framework for scheduling problems with controllable processing time and solved it using an algorithm for general linear assignment in $O(n^3)$ time. The unified objective function above is also a generalization of scheduling functions studied in [30].

We will use and extend the standard three-field notation $A|B|C$ introduced by Graham et al. [31] for scheduling problems. The A field describes the machine environment. For example, if 1 appears in the A field, it means that we deal with a single-machine scheduling problem. The B field defines the job-processing characteristics and constraints. The C field contains the optimizing criteria (for ease of presentation, shown in a symbolic fashion without including the summation indices).

The paper is organized as follows. In Section 2, we present an $O(n^4)$ optimization algorithm to determine the optimal schedule for the unified-problem. In Section 3, we present a group of scheduling problems, where the objective function includes costs for the total weighted number of tardy jobs, makespan, and due date assignment. We show that, for three different due date assignment methods, the objective can be reformulated as a special case of (4), which enables us to solve all these problems in $O(n^4)$ time. Using dynamic programming techniques, we improve this time complexity to be $O(n^2)$ for some important special cases in Section 4. The last section contains a summary and our concluding remarks.

2. The Unified Problem

We show in this section that our problem can be solved in $O(n^4)$ time by solving a linear assignment problem. First we consider the case when l is given. For $1 \leq i, j \leq n$, let us define c_{ij} as

$$c_{ij} = \begin{cases} \xi_i \eta_j & \text{for } i \leq l \\ \xi_i \eta_j + \psi_j & \text{for } l+1 \leq i \leq n. \end{cases} \quad (5)$$

Step 1. Calculate the c_{ij} values by (5).

Step 2. Solve the assignment problem ($P1(l)$) to determine the optimal job sequence, and denote the resulting optimal sequence by $\pi^* = ([1], [2], \dots, [n])$.

ALGORITHM 1: The optimization algorithm for the solution of the $1|| \sum \xi_j \eta_{[j]} + \sum \psi_{[j]}$ problem.

Initialization: $Z^* = \infty, l = 0$.

while $l \leq n$ **do**

Step 1. Apply Algorithm 1. Denote the optimal job sequence by $\pi^*(l) = ([1], [2], \dots, [n])$ and the minimum cost by $Z^*(l)$.

Step 2. If $Z^*(l) \leq Z^*$, then set $Z^* = Z^*(l), l^* = l, \pi^* = \pi^*(l)$.

Step 3. $l = l + 1$.

end

ALGORITHM 2

Our sequencing problem then reduces to the classical linear assignment problem of finding π which minimizes $\sum_{i=1}^n c_{i[i]}$. Since the calculation of the assignment costs depends on l , we denote this assignment problem by $P1(l)$. It is well known that a linear assignment problem can be solved in $O(n^3)$ time (see [32]).

Therefore, we have the following result.

Lemma 1. *The optimal sequence, denoted by π^* , can be obtained by solving a linear assignment problem requiring $O(n^3)$ time for fixed l .*

The results of our analysis are summarized as shown in Algorithm 1.

If l is not known, then we have to enumerate all l and run Algorithm 1 repeatedly as a subroutine (see Algorithm 2).

From Lemma 1, we can get the following result easily.

Theorem 2. *Algorithm 2 solves the $1|| \sum \xi_j \eta_{[j]} + \sum \psi_{[j]}$ problem in $O(n^4)$ time.*

In the next section, we present various applications of our general approach to solve a large set of important scheduling problems with variable processing times.

3. Applications to Solve Scheduling Problems

Let $C_{[k]}$ be the completion time of the k th job in the processing sequence whose processing time varies according to (3). The following crucial equality is given in [23].

Lemma 3. *Assuming that the jobs are processed from time zero and there is no idle time between them, then*

$$C_{[k]} = \sum_{j=1}^k \left(j^c \prod_{i=j+1}^k (1 + b i^c) \right) a_{[j]}, \quad (6)$$

where $\prod_{i=k+1}^k (1 + b(k+1)^c) = 1$ by definition.

Define

$$f_{jk} = j^c \prod_{i=j+1}^k (1 + b i^c) \quad (7)$$

for $j = 1, \dots, n$ and $k = j + 1, \dots, n$. Then (6) above can be rewritten as

$$C_{[k]} = \sum_{j=1}^k f_{jk} a_{[j]}. \quad (8)$$

3.1. Minimizing the Total Weighted Number of Tardy Jobs with Due Date Assignment. In this section, we show how our unified method can be used to minimize the total weighted number of tardy jobs with due date assignment decisions and variable processing times.

Since the solution of due date assignment problems also includes the determination of the due dates, a schedule for these problems is defined by a job sequence $\pi = ([1], [2], \dots, [n])$ and a due date assignment vector $\mathbf{d} = (d_1, d_2, \dots, d_n)$, and our objective is to determine the schedule which minimizes various objective functions defined in the subsections below. Therefore, to also show their dependence on the due dates, we redefine $g(\pi)$ from (4) as $g(\pi, \mathbf{d})$. Cases will be considered by providing special forms of $g(\pi, \mathbf{d})$.

Our objective is to find a job sequence $\pi^* = ([1], [2], \dots, [n])$ and a set of due dates $\mathbf{d}^* = (d_1^*, d_2^*, \dots, d_n^*)$ to minimize a cost function that includes the costs of tardiness, due date assignment, and makespan. For the CON, SLK, and DIF due date assignment methods, it is given by

$$g(\pi, \mathbf{d}) = \sum_{j=1}^n \psi_{[j]} U_{[j]} + \gamma \sum_{j=1}^n d_{[j]} + \delta C_{\max}, \quad (9)$$

where γ is a nonnegative parameter representing the cost of one unit of due date, δ is a nonnegative parameter representing the cost of one unit of makespan, $\psi_{[j]}$ is the cost of job $[j]$ being tardy, and $U_{[j]}$ is the tardiness indicator variable; that is, $U_{[j]} = 1$ if $C_{[j]} > d_{[j]}$ and 0 otherwise.

It is easy to see that the jobs are processed from time zero and there is no idle time between them; that is, the condition of Lemma 3 is satisfied for all the due date assignment methods. Therefore, (6) holds in all cases. Next we show that, under an optimal due date assignment strategy, the scheduling cost can be reduced to the format of (4) for all of them. We present results with different due date assignment methods, including CON, DIF, and SLK, to minimize total weighted number of tardy jobs.

In the following subsections, we will reduce the scheduling problem for each due date assignment method to the format of (4), which means that they can all be solved in $O(n^4)$ time. For any sequence π , let us define set $E(\pi)$ as the set of early (or on-time) jobs under an optimal due date assignment strategy and define the set of tardy jobs by $T(\pi) = J \setminus E(\pi)$. Then the following lemma from [33] is applicable to our problems (we omit the proof since it is rather straightforward).

Lemma 4. *There is an optimal schedule in which the corresponding π sequences $E(\pi)$ before set $T(\pi)$.*

Let $l = |E(\pi)|$ denote the number of early jobs in the optimal schedule, and then we have $E(\pi) = \{[1], [2], \dots, [l]\}$.

DIF Due Date Assignment Method. By Lemma 4, we can schedule the l early jobs before all the tardy jobs. For job j , if it is early, its due date is at least as large as its completion time. Therefore, the optimal assignment strategy is to let $d_{[j]}^* = C_{[j]}$, for $j = 1, \dots, l$. On the other hand, if job j is tardy, then we can set its due date as early as possible; that is, let $d_{[j]}^* = 0$ for $j = l+1, \dots, n$, since the job tardiness cost does not depend on the amount of tardiness. From (8) and (9), we have

$$\begin{aligned} g(\pi) &= \sum_{j=l+1}^n \psi_{[j]} + \gamma \sum_{k=1}^l C_{[k]} + \delta C_{[n]} \\ &= \sum_{j=l+1}^n \psi_{[j]} + \gamma \sum_{k=1}^l \sum_{j=1}^k f_{jk} a_{[j]} + \delta \sum_{j=1}^n f_{jn} a_{[j]} \\ &= \sum_{j=1}^l \left(\gamma \sum_{k=j}^l f_{jk} + \delta f_{jn} \right) a_{[j]} + \delta \sum_{j=l+1}^n f_{jn} a_{[j]} + \sum_{j=l+1}^n \psi_{[j]}. \end{aligned} \quad (10)$$

We can see that this is indeed in the format of (4), where

$$\xi_j = \begin{cases} \gamma \sum_{k=j}^l f_{jk} + \delta f_{jn} & \text{for } j \leq l \\ \delta f_{jn} & \text{for } l+1 \leq j \leq n. \end{cases} \quad (11)$$

This equation becomes simpler if there is only learning effect or only time-dependent processing times. If there is only learning effect, we have $b = 0$; therefore f_{jk} in (7) is reduced to j^c for every k . Thus the equation above becomes

$$\xi_j = \begin{cases} \gamma(l-j+1)j^c + \delta j^c & \text{for } j \leq l \\ \delta j^c & \text{for } l+1 \leq j \leq n. \end{cases} \quad (12)$$

On the other hand, if there is only time-dependent processing time effect, we have $c = 0$; therefore, f_{jk} becomes $(1+b)^{k-j}$. Thus the equation above becomes

$$\xi_j = \begin{cases} \gamma(l-j+1) \sum_{k=j}^l (1+b)^{k-j} \\ \quad + \delta(1+b)^{n-j} & \text{for } j \leq l \\ \delta(1+b)^{n-j} & \text{for } l+1 \leq j \leq n. \end{cases} \quad (13)$$

CON Due Date Assignment Method. It can be easily seen that the optimal strategy is to assign $C_{[l]}$ as the common due date $d_{[j]}^* = C_{[l]}$ for all j . From (8) and (9), we have

$$\begin{aligned} g(\pi) &= \sum_{j=l+1}^n \psi_{[j]} + n\gamma C_{[l]} + \delta C_{[n]} \\ &= \sum_{j=l+1}^n \psi_{[j]} + n\gamma \sum_{j=1}^l f_{jl} a_{[j]} + \delta \sum_{j=1}^n f_{jn} a_{[j]} \\ &= \sum_{j=1}^l (n\gamma f_{jl} + \delta f_{jn}) a_{[j]} + \sum_{j=l+1}^n \psi_{[j]} + \delta \sum_{j=l+1}^n f_{jn} a_{[j]}. \end{aligned} \quad (14)$$

We can see that this is indeed the format of (4), where

$$\xi_j = \begin{cases} n\gamma f_{jl} + \delta f_{jn} & \text{for } j \leq l \\ \delta f_{jn} & \text{for } l+1 \leq j \leq n. \end{cases} \quad (15)$$

If there is only learning effect, we have $b = 0$; therefore f_{jk} is reduced to j^c for every k . Thus the equation above becomes

$$\xi_j = \begin{cases} n\gamma j^c + \delta j^c & \text{for } j \leq l \\ \delta j^c & \text{for } l+1 \leq j \leq n. \end{cases} \quad (16)$$

On the other hand, if there is only time-dependent processing time effect, we have $c = 0$; therefore, f_{jk} becomes $(1+b)^{k-j}$. Thus the equation above becomes

$$\xi_j = \begin{cases} n\gamma(1+b)^{l-j} + \delta(1+b)^{n-j} & \text{for } j \leq l \\ \delta(1+b)^{n-j} & \text{for } l+1 \leq j \leq n. \end{cases} \quad (17)$$

SLK Due Date Assignment Method. Let $h = l - 1$. We know from [34] that the optimal common slack value is $s^* = C_{[h]}$. From (8) and (9), we have

$$\begin{aligned} g(\pi) &= \sum_{j=l+1}^n \psi_{[j]} + \gamma \sum_{j=1}^n (s^* + p_{[j]}) + \delta C_{[n]} \\ &= \sum_{j=l+1}^n \psi_{[j]} + n\gamma C_{[h]} + \gamma C_{[n]} + \delta C_{[n]} \\ &= \sum_{j=l+1}^n \psi_{[j]} + n\gamma \sum_{j=1}^h f_{jh} a_{[j]} + (\gamma + \delta) \sum_{j=1}^n f_{jn} a_{[j]} \end{aligned}$$

- Step 1.* Apply Algorithm 2 where ξ_j is calculated by (15) for the CON method, by (19) for the SLK method, and by (11) for the DIF method.
- Step 2.* For the CON method, assign $C_{[l^*]}$ to all $d_{[j]}^*$. For the SLK method, assign slack value $s^* = C_{[l^*-1]}$ to all jobs. For the DIF method, let $d_{[j]}^* = C_{[j]}$ for $j = 1, 2, \dots, l^*$, and $d_{[j]}^* = 0$ for $j = l^* + 1, \dots, n$.

ALGORITHM 3: A unified optimization algorithm for solving the $1 \mid p_{[j]} = (a_{[j]} + bS_{[j]})j^c, X \mid \sum_{j=1}^n \psi_j U_j + \gamma \sum_{j=1}^n d_j + \delta C_{\max}$ problems for $X \in \{\text{CON}, \text{SLK}, \text{DIF}\}$.

$$\begin{aligned}
 &= \sum_{j=1}^h (n\gamma f_{jh} + (\gamma + \delta) f_{jn}) a_{[j]} \\
 &+ (\gamma + \delta) \sum_{j=h+1}^n f_{jn} a_{[j]} + \sum_{j=l+1}^n \psi_{[j]}.
 \end{aligned} \tag{18}$$

We can see that this is indeed in the format of (4), where

$$\xi_j = \begin{cases} n\gamma f_{jh} + (\gamma + \delta) f_{jn} & \text{for } j \leq h \\ (\gamma + \delta) f_{jn} & \text{for } h+1 \leq j \leq n. \end{cases} \tag{19}$$

If there is only learning effect, we have $b = 0$; therefore f_{jk} is reduced to j^c . Thus the equation above becomes

$$\xi_j = \begin{cases} n\gamma j^c + (\gamma + \delta) j^c & \text{for } j \leq h \\ (\gamma + \delta) j^c & \text{for } h+1 \leq j \leq n. \end{cases} \tag{20}$$

On the other hand, if there is only time-dependent processing time effect, we have $c = 0$; therefore, f_{jk} becomes $(1+b)^{k-j}$. Thus the equation above becomes

$$\xi_j = \begin{cases} n\gamma(1+b)^{h-j} + (\gamma + \delta)(1+b)^{n-j} & \text{for } j \leq h \\ (\gamma + \delta)(1+b)^{n-j} & \text{for } h+1 \leq j \leq n. \end{cases} \tag{21}$$

Application of the Unified Optimization Algorithm. According to the analysis above, the objective function (9) can be written in the format of (4) for all three due date assignment methods, and thus we can present the following optimization algorithm to solve the problem for the weighted number of tardy jobs objective (see Algorithm 3).

By Theorem 2, we have the following result for Algorithm 3.

Theorem 5. Algorithm 3 solves the $1 \mid p_{[j]} = (a_{[j]} + bS_{[j]})j^c, X \mid \sum_{j=1}^n \psi_j U_j + \gamma \sum_{j=1}^n d_j + \delta C_{\max}$ problems for $X \in \{\text{CON}, \text{SLK}, \text{DIF}\}$ in $O(n^4)$ time.

It is easy to verify that if, instead of the makespan, we consider the total completion time in the objective of Theorem 5, that is, we replace $\delta C_{\max}$ with $\theta \sum C_j$, the same arguments hold. The only difference is that, in the definition of ξ_j , we use $\theta \sum_{k=j}^n f_{jk}$ instead of δf_{jn} . Therefore, we also state the following result.

Theorem 6. The $1 \mid p_{[j]} = (a_{[j]} + bS_{[j]})j^c, X \mid \sum_{j=1}^n \psi_j U_j + \gamma \sum_{j=1}^n d_j + \theta \sum C_j$ problems for $X \in \{\text{CON}, \text{SLK}, \text{DIF}\}$ are solvable in $O(n^4)$ time.

In the next two sections, we omit the makespan considerations; that is, $\delta = 0$. The objective function (9) now becomes

$$g(\pi) = \sum_{j=1}^n \psi_j U_j + \gamma \sum_{j=1}^n d_j. \tag{22}$$

3.2. Faster Algorithms for Cases with Time-Dependent Processing Times Only. In this section, we present quadratic-time algorithms for cases without learning effect ($c = 0$ in (3)) or makespan considerations ($\delta = 0$) using dynamic programming techniques. In this case (7) becomes

$$f_{jk} = (1+b)^{k-j}. \tag{23}$$

Our method is to give the recursive relation for the optimal scheduling cost, which contains a term $a_{[j]}$ times, some coefficient. To this end, we need to rewrite $C_{[k]}$ in (8) and $\sum_{j=1}^k C_{[j]}$ in terms of the $a_{[j]}$'s as follows:

$$C_{[k]} = \sum_{j=1}^k (1+b)^{k-j} a_{[j]}. \tag{24}$$

For the ease of the following summation calculation, we change it to the following form:

$$C_{[j]} = \sum_{i=1}^j (1+b)^{j-i} a_{[i]}, \tag{25}$$

$$\begin{aligned}
 \sum_{j=1}^k C_{[j]} &= \sum_{j=1}^k \sum_{i=1}^j f_{ij} a_{[i]} \\
 &= \sum_{j=1}^k \sum_{i=j}^k f_{ji} a_{[j]} \\
 &= \sum_{j=1}^k \sum_{i=j}^k (1+b)^{i-j} a_{[j]} \\
 &= \sum_{j=1}^k \frac{(1+b)^{k-j+1} - 1}{b} a_{[j]},
 \end{aligned} \tag{26}$$

where we changed the order of summation for the second equality.

Since the per-unit due date assignment cost γ is the same for each job j in (22) and the order of tardy jobs is immaterial, we can assume that the early jobs are always sequenced in SPT order. Thus, we sort the jobs into SPT order and decide which job is to be early or tardy in a backward fashion. For the sake of simplicity, suppose they are already sorted in this way.

Notice that in (24) and (26), the coefficient of $a_{[j]}$ is dependent only on the number of early jobs after job $[j] : k-j$. In fact, a_j has a “weight” of $(1+b)^{k-j}$ (in other words, it contributes $(1+b)^{k-j}$ “times”) in the calculation of $C_{[k]}$, and it has a “weight” of $((1+b)^{k-j+1} - 1)/b$ (in other words, it contributes $((1+b)^{k-j+1} - 1)/b$ “times”) in the summation $\sum_{j=1}^k C_{[j]}$.

This enables us to design the following backward dynamic programming. Define $w(j, k)$ to be the minimum cost of scheduling jobs $j, j+1, \dots, n$ s.t. k of them are early, for $1 \leq j \leq n$ and $0 \leq k \leq n-j+1$. We have the following set of boundary conditions for all cases when all jobs are tardy:

$$w(j, 0) = \sum_{i=j}^n \psi_{[i]}, \quad (27)$$

and another boundary condition when $j = n$:

$$w(n, 1) = \gamma a_n. \quad (28)$$

Now we describe the recursive functions to calculate $w(j, k)$ for the three due date assignment methods.

DIF. If job j is early, then there will be $k-1$ early jobs among $j+1, \dots, n$. The optimal due date for job j is C_j . By (22) and (26), a_j is counted $\gamma((1+b)^k - 1)/b$ “times.” Therefore

$$w(j, k) = w(j+1, k-1) + \gamma \frac{(1+b)^k - 1}{b} a_j; \quad (29)$$

otherwise, when job j is tardy, the optimal due date for job j is 0; thus

$$w(j, k) = w(j+1, k) + \psi_j. \quad (30)$$

Therefore, we have the following recursive equation:

$$w(j, k) = \min \left\{ w(j+1, k-1) + \gamma \frac{(1+b)^k - 1}{b} a_j, w(j+1, k) + \psi_j \right\}. \quad (31)$$

CON. By the due date assignment rule, each job is assigned the same due date, which is the sum of the processing times of all the early jobs. Therefore, if job j is early, then it will be counted n times. Thus

$$w(j, k) = w(j+1, k-1) + n\gamma(1+b)^{k-1} a_j \quad (32)$$

by (22) and (24) and the fact that there are $k-1$ early jobs after job j ; otherwise, when job j is tardy,

$$w(j, k) = w(j+1, k) + \psi_j. \quad (33)$$

Therefore, we have the following recursive equation:

$$w(j, k) = \min \left\{ w(j+1, k-1) + n\gamma(1+b)^{k-1} a_j, w(j+1, k) + \psi_j \right\}. \quad (34)$$

SLK. From [34], we know that the optimal common slack value is $s^* = C_{[l-1]}$, where l is the number of early jobs. In other words, the last early job does not contribute to the due date assignment cost, but the other ones (the first $l-1$) do. If job j is early and $k \geq 2$, then

$$w(j, k) = w(j+1, k-1) + n\gamma(1+b)^{k-1} a_j \quad (35)$$

by (22) and (24) and the fact that there are $k-1 > 0$ early jobs after job j ; if job j is early and $k = 1$, then it is the last early job and thus

$$w(j, k) = w(j+1, k-1); \quad (36)$$

that is,

$$w(j, 1) = w(j+1, 0) = \sum_{i=j+1}^n \psi_{[i]}; \quad (37)$$

if job j is tardy, we have

$$w(j, k) = w(j+1, k) + \psi_j. \quad (38)$$

Therefore, we have the following recursive equation:

$$w(j, k) = \min \left\{ w(j+1, k-1) + n\gamma(1+b)^{k-1} a_j, w(j+1, k) + \psi_j \right\}, \quad (39)$$

for $k \geq 2$, and (for $k = 1$)

$$w(j, 1) = \min \left\{ w(j+1, 0), w(j+1, 1) + \psi_j \right\} = \min \left\{ \sum_{i=j+1}^n \psi_{[i]}, w(j+1, 1) + \psi_j \right\}. \quad (40)$$

From the analysis above, we can present Algorithms 4 and 5.

In summary, we have the following results.

Theorem 7. Algorithm 4 solves the 1 | $p_{[j]} = a_{[j]} + bS_{[j]}, X$ | $\sum_{j=1}^n \psi_j U_j + \gamma \sum_{j=1}^n d_j$ problems for $X \in \{\text{CON}, \text{DIF}\}$ in $O(n^2)$ time. Algorithm 5 solves the 1 | $p_{[j]} = a_{[j]} + bS_{[j]}, \text{SLK}$ | $\sum_{j=1}^n \psi_j U_j + \gamma \sum_{j=1}^n d_j$ problem in $O(n^2)$ time.

Proof. The correctness of the algorithms follows from (24)–(40) and the preceding discussion. For the time complexity, we note that, in both algorithms, the outer recursion j takes $O(n)$ time, and the inner recursion on k also takes $O(n)$ time. Therefore, the total time complexity is $O(n^2)$. $\square$

Initialization:

$$w(n, 1) := \gamma a_n$$

For $j := 1$ **To** n **Do**

$$w(j, 0) := \sum_{i=j}^n \psi_{[i]}$$

Recursion:

For $j := n - 1$ **Down To** 1 **Do**

For $k := 1$ **To** $n - j + 1$ **Do**

for CON:

$$w(j, k) = \min\{w(j + 1, k - 1) + n\gamma(1 + b)^{k-1}a_j, w(j + 1, k) + \psi_j\}$$

for DIF:

$$w(j, k) = \min\left\{w(j + 1, k - 1) + \gamma \frac{(1 + b)^k - 1}{b} a_j, w(j + 1, k) + \psi_j\right\}$$

Optimal solution value:

$$\min\{w(1, k) \mid 0 \leq k \leq n\}.$$

End

ALGORITHM 4: The optimization algorithm for the solution of 1 | $p_{[j]} = a_{[j]} + bS_{[j]}$, $X \mid \sum_{j=1}^n \psi_j U_j + \gamma \sum_{j=1}^n d_j$ problems for $X \in \{\text{CON}, \text{DIF}\}$.

Initialization:

$$w(n, 1) := \gamma a_n$$

For $j := 1$ **To** n **Do**

$$w(j, 0) := \sum_{i=j}^n \psi_{[i]}$$

Recursion:

For $j := n - 1$ **Down To** 1 **Do**

$$w(j, 1) = \min\{w(j + 1, 0), w(j + 1, 1) + \psi_j\} = \min\left\{\sum_{i=j+1}^n \psi_{[i]}, w(j + 1, 1) + \psi_j\right\}$$

For $j := n - 1$ **Down To** 1 **Do**

For $k := 2$ **To** $n - j + 1$ **Do**

$$w(j, k) = \min\{w(j + 1, k - 1) + n\gamma(1 + b)^{k-1}a_j, w(j + 1, k) + \psi_j\}$$

Optimal solution value:

$$\min\{w(1, k) \mid 0 \leq k \leq n\}.$$

End

ALGORITHM 5: The optimization algorithm for the solution of 1 | $p_{[j]} = a_{[j]} + bS_{[j]}$, SLK | $\sum_{j=1}^n \psi_j U_j + \gamma \sum_{j=1}^n d_j$ problem.

3.3. Faster Algorithms for Cases with Learning Effect Only. In this section, we present quadratic-time algorithms for cases without time-dependent effect ($b = 0$ in (3)), or makespan consideration ($\delta = 0$), using forward dynamic programming techniques. In this case, we have from (7)

$$f_{jk} = f^c. \quad (41)$$

As in the previous section, our method is to give the recursive relation for the optimal scheduling costs, which contain the term $a_{[j]}$ times, some coefficient. To this end, we need to rewrite $C_{[k]}$ in (8) in terms of $a_{[j]}$ as follows:

$$C_{[k]} = \sum_{j=1}^k f^c a_{[j]}. \quad (42)$$

Similarly to the discussion in the previous section, we can assume that the early jobs are already sequenced in SPT order.

Our goal is to decide which job is to be early or tardy in a forward fashion this time.

Notice that, in (42), the coefficient of $a_{[j]}$ is dependent only on the number of early jobs before job $[j]$. This enables us to design the following forward dynamic programming algorithms.

CON. Define $w(j, k)$ to be the minimum cost of scheduling jobs $1, 2, \dots, j$ s.t. k of them are early, for $1 \leq k \leq j \leq n$. We have the following set of boundary conditions when jobs $1, 2, \dots, j$ are all tardy:

$$w(j, 0) = \sum_{i=1}^j \psi_{[i]}, \quad (43)$$

for $1 \leq j \leq n$, and

$$w(1, 1) = a_1. \quad (44)$$

Initialization:	$w(1, 1) := \gamma a_1$
For $j := 1$ To n Do	$w(j, 0) := \sum_{i=1}^j \psi_i$
Recursion:	
For $j := 1$ To n Do	
For $k := 0$ To j Do	$w(j, k) = \min\{w(j-1, k-1) + nj^c a_j, w(j-1, k) + \psi_j\}$
	$W(j) = \min_{0 \leq k \leq j-1} w(j-1, k) + \sum_{i=j+1}^n \psi_i$
Optimal solution value:	
For CON:	$\min\{w(1, k) \mid 0 \leq k \leq n\}.$
For SLK:	$W = \min_{1 \leq j \leq n} W(j).$
End	

ALGORITHM 6: The optimization algorithm for the solution of $1 \mid p_{[j]} = a_{[j]} j^c, X \mid \sum_{j=1}^n \psi_j U_j + \gamma \sum_{j=1}^n d_j$ problems for $X \in \{\text{CON}, \text{SLK}\}$.

If job j is early, then its processing time gets counted in the (common) due date of every job, that is, n times, and thus

$$w(j, k) = w(j-1, k-1) + nj^c a_j \quad (45)$$

by (22) and (42) and the fact that there are $k-1$ early jobs before job j ; otherwise, job j is tardy, and we have

$$w(j, k) = w(j-1, k) + \psi_j. \quad (46)$$

Therefore, we have the following recursive equation:

$$w(j, k) = \min\{w(j-1, k-1) + nj^c a_j, w(j-1, k) + \psi_j\}. \quad (47)$$

SLK. As in the previous section, we know from [34] that the optimal common slack value is $s^* = C_{[l-1]}$, where l is the number of early jobs. In other words, the last early job does not contribute to the due date assignment cost, but the other ones (the first $l-1$) do. Due to this special property, we do not have the same direct recursive equations as in the CON case. However, we can still utilize those recursive equations to get a set of recursive equations for the SLK case.

Let $W(r)$ be the minimum cost to schedule jobs of $1, 2, \dots, n$ s.t. job r is the last early one. By the above discussion and definition of $w(j, k)$ in the CON part (in which the due date assignment cost for the k early jobs is calculated recursively), we have

$$W(r) = \min_{0 \leq k \leq r-1} w(r-1, k) + \sum_{i=r+1}^n \psi_i \quad (48)$$

for $1 \leq r \leq n$. We define $w(0, 0) = 0$ and the minimum cost to schedule jobs $1, 2, \dots, n$ is given by

$$W = \min_{1 \leq r \leq n} W(r). \quad (49)$$

From the analysis above, we are ready to present Algorithm 6.

In summary, we have the following result.

Theorem 8. Algorithm 6 solves the $1 \mid p_{[j]} = a_{[j]} j^c, X \mid \sum_{j=1}^n \psi_j U_j + \gamma \sum_{j=1}^n d_j$ problems for $X \in \{\text{CON}, \text{SLK}\}$ in $O(n^2)$ time.

Proof. The correctness of the algorithms follows from the analysis before them. For the time complexity, we note that, in both algorithms, the outer recursion j takes $O(n)$ time, and the inner recursion k also takes $O(n)$ time. Therefore, the total time complexity is $O(n^2)$. $\square$

4. Summary and Concluding Remarks

We considered single machine scheduling problems with learning/deterioration effects and time-dependent processing times, with due date assignment considerations and with the objective to minimize the weighted number of tardy jobs. We gave polynomial-time solutions for all problems considered. It remains an open question whether there exists a $O(n^2)$ time algorithm for the DIF due date assignment method with learning effect only. Another challenge is to design a $O(n^2)$ time algorithm for the case of $\delta > 0$, that is, to consider the makespan.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This research was supported in part by Discovery Grant 1798-08 by the Natural Sciences and Engineering Research Council of Canada.

References

- [1] T. P. Wright, "Factors affecting the cost of airplanes," *Journal of Aeronautical Sciences*, vol. 3, pp. 122–128, 1936.
- [2] G. K. Webb, "Integrated circuit (IC) pricing," *Journal of High Technology Management Research*, vol. 5, no. 2, pp. 247–260, 1994.
- [3] C. Sáenz-Royo and V. Salas-Fumás, "Learning to learn and productivity growth: evidence from a new car-assembly plant," *Omega*, vol. 41, no. 2, pp. 336–344, 2013.
- [4] S. Gawiejnowicz, "A note on scheduling on a single processor with speed dependent on a number of executed jobs," *Information Processing Letters*, vol. 57, no. 6, pp. 297–300, 1996.
- [5] D. Biskup, "Single-machine scheduling with learning considerations," *European Journal of Operational Research*, vol. 115, no. 1, pp. 173–178, 1999.
- [6] V. S. Gordon, C. N. Potts, V. A. Strusevich, and J. D. Whitehead, "Single machine scheduling models with deterioration and learning: handling precedence constraints via priority generation," *Journal of Scheduling*, vol. 11, no. 5, pp. 357–370, 2008.
- [7] G. Mosheiov, "Parallel machine scheduling with a learning effect," *Journal of the Operational Research Society*, vol. 52, no. 10, pp. 1165–1169, 2001.
- [8] G. Mosheiov and J. B. Sidney, "Note on scheduling with general learning curves to minimize the number of tardy jobs," *Journal of the Operational Research Society*, vol. 56, no. 1, pp. 110–112, 2005.
- [9] D. Okołoski and S. Gawiejnowicz, "Exact and heuristic algorithms for parallel-machine scheduling with de Jong's learning effect," *Computers and Industrial Engineering*, vol. 59, no. 2, pp. 272–279, 2010.
- [10] 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.
- [11] A. Janiak and R. Rudek, "Scheduling problems with position dependent job processing times," in *Scheduling in Computer and Manufacturing Systems*, A. Janiak, Ed., pp. 26–38, Wydawnictwa Komunikacji i Łączności, Wrocław, Poland, 2006.
- [12] A. Janiak, T. Krysiak, and R. Trela, "Scheduling problems with learning and ageing effects: a survey," *Decision Making in Manufacturing and Services*, vol. 5, no. 1-2, pp. 19–36, 2011.
- [13] K. Rustogi and V. A. Strusevich, "Simple matching vs linear assignment in scheduling models with positional effects: a critical review," *European Journal of Operational Research*, vol. 222, no. 3, pp. 393–407, 2012.
- [14] S. K. Gupta, A. S. Kunnathur, and K. Dandapani, "Optimal repayment policies for multiple loans," *Omega*, vol. 15, no. 4, pp. 323–330, 1987.
- [15] N. P. Rachaniotis and C. P. Pappis, "Scheduling fire-fighting tasks using the concept of 'deteriorating jobs'," *Canadian Journal of Forest Research*, vol. 36, no. 3, pp. 652–658, 2006.
- [16] A. S. Kunnathur and S. K. Gupta, "Minimizing the makespan with late start penalties added to processing times in a single facility scheduling problem," *European Journal of Operational Research*, vol. 47, no. 1, pp. 56–64, 1990.
- [17] G. Mosheiov, "Λ-shaped policies to schedule deteriorating jobs," *Journal of the Operational Research Society*, vol. 47, no. 9, pp. 1184–1191, 1996.
- [18] S. Browne and U. Yechiali, "Scheduling deteriorating jobs on a single processor," *Operations Research*, vol. 38, no. 3, pp. 495–498, 1990.
- [19] G. Mosheiov, "V-shaped policies for scheduling deteriorating jobs," *Operations Research*, vol. 39, no. 6, pp. 979–991, 1991.
- [20] T. C. E. Cheng, Q. Ding, and B. M. T. Lin, "A concise survey of scheduling with time-dependent processing times," *European Journal of Operational Research*, vol. 152, no. 1, pp. 1–13, 2004.
- [21] S. Gawiejnowicz, *Time-Dependent Scheduling*, Springer, Berlin, Germany, 2008.
- [22] W. C. Lee, "A note on deteriorating jobs and learning in single-machine scheduling problems," *International Journal of Business and Economics*, vol. 3, no. 1, pp. 83–89, 2004.
- [23] J.-B. Wang, "A note on scheduling problems with learning effect and deteriorating jobs," *International Journal of Systems Science*, vol. 37, no. 12, pp. 827–833, 2006.
- [24] J.-B. Wang, "Single-machine scheduling problems with the effects of learning and deterioration," *Omega*, vol. 35, no. 4, pp. 397–402, 2007.
- [25] X. Wang and T. C. E. Cheng, "Single-machine scheduling with deteriorating jobs and learning effects to minimize the makespan," *European Journal of Operational Research*, vol. 178, no. 1, pp. 57–70, 2007.
- [26] L. Y. Wang, J. B. Wang, W. J. Guo, X. Huang, and E. M. Feng, "Two single-machine scheduling problems with the effects of deterioration and learning," *International Journal of Advanced Manufacturing Technology*, vol. 46, no. 5–8, pp. 715–720, 2007.
- [27] J.-B. Wang and T. C. E. Cheng, "Scheduling problems with the effects of deterioration and learning," *Asia-Pacific Journal of Operational Research*, vol. 24, no. 2, pp. 245–261, 2007.
- [28] C. Koulamas, S. Gupta, and G. J. Kyparisis, "A unified analysis for the single-machine scheduling problem with controllable and non-controllable variable job processing times," *European Journal of Operational Research*, vol. 205, no. 2, pp. 479–482, 2010.
- [29] Y. Leyvand, D. Shabtay, and G. Steiner, "A unified approach for scheduling with convex resource consumption functions using positional penalties," *European Journal of Operational Research*, vol. 206, no. 2, pp. 301–312, 2010.
- [30] J. Qian and G. Steiner, "Fast algorithms for scheduling with learning effects and time-dependent processing times on a single machine," *European Journal of Operational Research*, vol. 225, no. 3, pp. 547–551, 2013.
- [31] 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.
- [32] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1982.
- [33] D. Shabtay and G. Steiner, "Two due date assignment problems in scheduling a single machine," *Operations Research Letters*, vol. 34, no. 6, pp. 683–691, 2006.
- [34] D. Shabtay and G. Steiner, "Optimal due date assignment and resource allocation to minimize the weighted number of tardy jobs on a single machine," *Manufacturing and Service Operations Management*, vol. 9, no. 3, pp. 332–350, 2007.