Single machine due-date scheduling of jobs with decreasing start-time dependent processing times

T.C.E. Cheng[1], L.Y. Kang[1,2], C.T. Ng[1]

[1]Department of Logistics, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong

[2]Department of Mathematics, Shanghai University, Shanghai 200436, China

Abstract

We study the problem of scheduling jobs whose processing times are decreasing functions of their starting times. We consider the case of a single machine and a common decreasing rate for the processing times. The problem is to determine an optimal combination of the due-date and schedule so as to minimize the sum of due date, earliness and tardiness penalties. We give an $O(n \log n)$ time algorithm to solve this problem.

Key words: Single machine scheduling; Due-date; Deteriorating jobs.

Introduction

Machine scheduling problems with start-time dependent job processing times have received increasing attention from the scheduling community in recent years [1, 2, 3, 6, 12, 13, 14]. Researchers have formulated different models for this phenomenon and solved different versions of the problem for various criteria. A survey of scheduling research with start-time dependent processing times can be found in Cheng et al. [7]. Generally, there are two groups

of models to describe this kind of scheduling processes. The first group is devoted to the problems in which the job processing times are characterized by non-decreasing functions of their starting times, and the second group concerns the problems in which the job processing times are non-increasing functions of their processing times. In this paper, we study the latter group of problems.

Application examples of the non-decreasing model of job processing time are quite intuitively different from its non-increasing counterpart. The latter model can be used to describe the process by which aerial threats are to be recognized by a radar station. In this case, a radar station has detected some objects approaching it. The time required to recognize the objects decreases as the objects get closer. Thus, the later the objects are detected, the smaller is the time for their recognition. Another example refers to the so-called 'learning effect'. Assume that a worker has to assemble a large number of similar products. The time required by the worker to assemble one product depends on his knowledge, skills, organization of his working place and others. The worker learns how to produce over time. After some time, he is better skilled, his working place is better organized and his knowledge is increased. As a result of his learning, the time required to assemble subsequent products decreases.

Problem formulation

There are given a single machine and a set $\mathcal{J} = \{1, 2, \ldots, n\}$ of n independent and non-preemptive jobs, which are immediately available for processing. The processing time p_i of job i is given as a linear decreasing function of its starting time s_i:

$$p_i(s_i) = a_i - bs_i,$$

where $a_i > 0$ denotes the normal processing time of job i and b denotes its decreasing rate, which is assumed to be common for all jobs. It is further

assumed that the decreasing rate b satisfies the following conditions:

$$0 < b < 1 \text{ and } b(\sum_{j=1}^{n} a_j - a_i) < a_i, \text{ for } 1 \leq i \leq n.$$

The first condition ensures that the decrease in job processing time is less than one unit for each unit of delay in its starting moment. The second condition ensures that all job processing times are positive in non-delay schedules. The paper only considers cases where non-delay schedules are optimal.

In addition, we cast the problem as the Common Due-Date Problem (CDDP), which deals with job scheduling on a single machine in a just-in-time production environment [4, 8, 9, 10, 14]. Prescribing a common due-date might represent a situation where several items constitute a single customer's order, or it might reflect an assembly environment in which the components should all be ready at the same time in order to avoid staging delays.

For any given schedule σ, let

$$\begin{aligned}
s_i(\sigma) &= \text{start time of job } i, \\
p_i(\sigma) &= a_i - bs_i(\sigma), \text{actual processing time of job } i, \\
C_i(\sigma) &= \text{completion time of job } i, \\
E_i(\sigma) &= \max\{0, d - C_i(\sigma)\}, \text{earliness of job } i, \\
T_i(\sigma) &= \max\{0, C_i(\sigma) - d\}, \text{tardiness of job } i, \\
f(d, \sigma) &= \sum(\alpha E_i(\sigma) + \beta T_i(\sigma) + \gamma d), \text{total penalty function, where } \alpha, \beta, \gamma \\
&\quad \text{are the unit earliness, tardiness and due-date penalty, respectively.}
\end{aligned}$$

In this paper, we consider the problem of finding the optimal combination of the schedule σ^* and the common due date d^* that leads to the minimum value of

$$f(d, \sigma) = \sum_{i=1}^{n}(\alpha E_i + \beta T_i + \gamma d).$$

Using the three field notation $\alpha|\beta|\gamma$ of Graham et al. [11], the problem can be denoted as $1|p_i(s_i) = a_i - bs_i, d|\sum_{i=1}^{n}(\alpha E_i + \beta T_i + \gamma d)$.

Preliminary analysis

We first present some elementary results. For any given schedule σ, we will use $\sigma(j)$ to denote the job in position j of σ, for $j = 1, \cdots, n$. Define $\sigma(0) = 0$ and $C_0 = 0$ The following property is easy to see.

Property 1. There exists an optimal schedule in which the machine is not idle between the processing of the jobs.

Similar to Theorem 7.2 in [5], we have

Property 2. For any specified schedule σ, there exists an optimal due-date $d^* = C_{\sigma(K)}$, where K is the smallest non-negative integer value greater than or equal to $\frac{n\beta - n\gamma}{\alpha+\beta}$, i.e., exactly K jobs are nontardy.

Proof. We first show that for any specified schedule σ, d^* is equal to the completion time of some job.

Consider a given schedule σ and d with $C_{\sigma(i)} < d < C_{\sigma(i+1)}$, and let F be the corresponding objective value. Define $x = d - C_{\sigma(i)}$ and $y = C_{\sigma(i+1)} - d$. Let F' and F'' be the objective value for $d = C_{\sigma(i)}$ and $d = C_{\sigma(i+1)}$, respectively. Then,

$$F' = F + x(n-i)\beta - xi\alpha - n\gamma x \tag{1}$$

and

$$F'' = F - y(n-i)\beta + yi\alpha + n\gamma y. \tag{2}$$

Thus, we have $F' \leq F$ if $(n-i)\beta \leq i\alpha + n\gamma$, and $F'' < F$ otherwise. This implies that for any specified schedule σ, d^* is equal to the completion time of some job $\sigma(k)$. Assume that $d^* = C_{\sigma(k)}$, and let Z be the optimal solution. It is easy to see that $d^* = C_{\sigma(0)}$ if $\beta \leq \gamma$. Otherwise, applying (1) to the situation $x = C_{\sigma(k)} - C_{\sigma(k-1)}$, we have $k(\alpha+\beta) \leq n\beta - n\gamma$, a contradiction. If $\beta > \gamma$, applying (1) and (2) to the situation $x = C_{\sigma(k)} - C_{\sigma(k-1)}$ and $y = C_{\sigma(k+1)} - C_{\sigma(k)}$, respectively, we conclude that $\frac{n\beta - n\gamma}{\alpha+\beta} \leq k \leq \frac{n\beta - n\gamma}{\alpha+\beta} + 1$. This implies that $d^* = C_{\sigma(K)}$. The result follows. ■

Due to Properties 1 and 2, we see that the total penalty for any given schedule σ is equal to $f(C_{\sigma(K)}, \sigma)$. Introducing $C_{\sigma(K)} = p_{\sigma(1)} + p_{\sigma(2)} + \ldots + p_{\sigma(K)}$, we get

$$f(C_{\sigma(K)}, \sigma) = \sum_{i=1}^{K} (\alpha(i-1) + n\gamma) p_{\sigma(i)} + \sum_{i=K+1}^{n} \beta(n+1-i) p_{\sigma(i)}. \tag{3}$$

For notational convenience, we define the following:

$$\begin{aligned}
m_i &= b \sum_{j=i}^{K} (\alpha(j-1) + n\gamma)(1-b)^{j-i} + b\beta \sum_{j=K+1}^{n} (n+1-j)(1-b)^{j-i}, \\
&\qquad \text{for } 2 \le i \le K, \qquad (4) \\
m_i &= b\beta \sum_{j=i}^{n} (n+1-j)(1-b)^{j-i}, \text{ for } K+1 \le i \le n, \qquad (5) \\
g_1(b) &= \alpha + (\alpha + n\gamma)b - bm_3, \\
g_2(b) &= \beta((n-(K+1))b - 1) - bm_{K+3}, \\
h_1(i) &= \alpha(i-1) + n\gamma - m_{i+1}, \text{ for } 1 \le i \le K-1, \\
h_2(i) &= (n+1-i)\beta - m_{i+1}, \text{ for } K \le i < n.
\end{aligned}$$

It is easy to see from (2) and (3) that

$$\begin{aligned}
m_{i+1} &= (\alpha i + n\gamma)b + (1-b)m_{i+2}, \qquad \text{for } 1 \le i \le K-1, \qquad (6) \\
m_{i+1} &= \beta(n-i)b + (1-b)m_{i+2}, \qquad \text{for } K \le i \le n-2. \qquad (7)
\end{aligned}$$

Property 3. (1). If $g_1(b) > 0$, then $\alpha + (\alpha i + n\gamma)b - bm_{i+2} > 0$, for $1 \le i \le K-1$.

(2). If $g_1(b) < 0$, then $\alpha + (\alpha i + n\gamma)b - bm_{i+2} < 0$, for $1 \le i \le K-1$.

(3). If $g_1(b) = 0$, then $\alpha + (\alpha i + n\gamma)b - bm_{i+2} = 0$, for $1 \le i \le K-1$.

Proof. (1). We proceed by induction on i. If $i = 1$, $\alpha + (\alpha i + n\gamma)b - bm_3 = g_1(b) > 0$, the result follows. Assume that the result holds for the case $i < k$. For the case $i = k$, by the induction hypothesis,

$$\alpha + (\alpha(k-1) + n\gamma)b - bm_{k+1} > 0. \tag{8}$$

Combining this with

$$m_{k+1} = (\alpha k + n\gamma)b + (1-b)m_{k+2}, \qquad \text{(By (4))} \tag{9}$$

we have

$$\begin{aligned}
& \alpha + (\alpha k + n\gamma)b - bm_{k+2} \\
= \;& \alpha + (\alpha k + n\gamma)b - \frac{bm_{k+1} - b^2(\alpha k + n\gamma)}{1-b} && \text{(By (7))} \\
> \;& \alpha + (\alpha k + n\gamma)b - \frac{\alpha + (\alpha(k-1) + n\gamma)b - b^2(\alpha k + n\gamma)}{1-b} && \text{(By (6))} \\
= \;& 0.
\end{aligned}$$

The result of (1) follows.

(2) and (3). Similar to the proof of (1).

Property 4. (1). If $g_1(b) > 0$, then for any optimal schedule σ, $a_{\sigma(1)} \geq a_{\sigma(2)} \geq \ldots \geq a_{\sigma(K)}$.

(2). If $g_1(b) \leq 0$, then there exists an optimal schedule σ such that $a_{\sigma(1)} \leq a_{\sigma(2)} \leq \ldots \leq a_{\sigma(K)}$.

Proof. (1). Assume that the schedule σ in which $a_{\sigma(i)} < a_{\sigma(i+1)}$ $(1 \leq i \leq K-1)$ is optimal. Let σ' be the schedule derived from σ by swapping i and $i+1$. Then,

$$\begin{aligned}
C_{\sigma(i+1)} &= a_{\sigma(i+1)} + (1-b)a_{\sigma(i)} + (1-b)^2 s_{\sigma(i)}, \\
C_{\sigma'(i+1)} &= a_{\sigma(i)} + (1-b)a_{\sigma(i+1)} + (1-b)^2 s_{\sigma(i)}.
\end{aligned}$$

So $C_{\sigma(i+1)} - C_{\sigma'(i+1)} = b(a_{\sigma(i+1)} - a_{\sigma(i)}) > 0$. Combining this with (1), the difference in the value of $f(d,s)$ between the two schedules is as follows:

$$\begin{aligned}
& f(C_{\sigma(K)}, \sigma) - f(C_{\sigma'(K)}, \sigma') \\
= \;& (\alpha(i-1) + n\gamma)p_{\sigma(i)} + (\alpha i + n\gamma)p_{\sigma(i+1)} \\
& -(\alpha(i-1) + n\gamma)p_{\sigma'(i)} + (\alpha i + n\gamma)p_{\sigma'(i+1)} - bm_{i+2}(a_{\sigma(i+1)} - a_{\sigma(i)}) \\
= \;& (\alpha(i-1) + n\gamma)(a_{\sigma(i)} - bs_{\sigma(i)}) + (\alpha i + n\gamma)(a_{\sigma(i+1)} - ba_{\sigma(i)} - b(1-b)s_{\sigma(i)}) \\
& -(\alpha(i-1) + n\gamma)(a_{\sigma(i+1)} - bs_{\sigma(i)}) - (\alpha i + n\gamma)(a_{\sigma(i)} - ba_{\sigma(i+1)} - b(1-b)s_{\sigma(i)}) \\
& -bm_{i+2}(a_{\sigma(i+1)} - a_{\sigma(i)}) \\
= \;& (\alpha + (\alpha i + n\gamma)b - bm_{i+2})(a_{\sigma(i+1)} - a_{\sigma(i)}) \\
> \;& 0, && \text{(By Property 3)}
\end{aligned}$$

a contradiction to the optimality of σ. This completes the proof of (1).

(2). Assume that the schedule σ_1 in which $a_{\sigma_1(i)} > a_{\sigma_1(i+1)}$ $(1 \leq i \leq K-1)$ is optimal. Let σ_1' be the schedule obtained from σ_1 by swapping i and $i+1$. Then,

$$\begin{aligned} & f(C_{\sigma_1(K)}, \sigma_1) - f(C_{\sigma_1'(K)}, \sigma_1') \\ = \; & (\alpha + (\alpha i + n\gamma)b - bm_{i+2})(a_{\sigma_1(i+1)} - a_{\sigma_1(i)}) \\ \geq \; & 0. \qquad\qquad \text{(By Property 3)} \end{aligned}$$

So, σ_1' is an optimal schedule. Proceeding as above, we can get an optimal schedule σ such that $a_{\sigma(1)} \leq a_{\sigma(2)} \leq \ldots \leq a_{\sigma(K)}$. ■

Property 5. (1). If $g_2(b) < 0$, then $\beta((n-i)b-1) - bm_{i+2} < 0$ for $K+1 \leq i < n$.

(2). If $g_2(b) > 0$, then $\beta((n-i)b-1) - bm_{i+2} > 0$ for $K+1 \leq i < n$.

(3). If $g_2(b) = 0$, then $\beta((n-i)b-1) - bm_{i+2} = 0$ for $K+1 \leq i < n$.

Proof. (1). We proceed by induction on i. If $i = K+1$, $\beta((n-i)b-1) - bm_{i+2} = g_2(b) < 0$, the result follows. Assume that the result holds for the case $K+1 \leq i < k < n$. We consider the case $i = k$. By the induction hypothesis,

$$\beta((n-k+1)b-1) - bm_{k+1} < 0. \tag{10}$$

Combining this with

$$bm_{k+1} = \beta(n-k)b^2 + (1-b)bm_{k+2}, \qquad \text{(By (5))} \tag{11}$$

we have

$$\begin{aligned} & \beta((n-k)b-1) - bm_{k+2} \\ = \; & \beta((n-k)b-1) - \frac{bm_{k+1} - \beta b^2(n-k)}{1-b} \qquad \text{(By (9))} \\ < \; & \beta((n-k)b-1) - \frac{\beta((n-k+1)b-1) - \beta b^2(n-k)}{1-b} \qquad \text{(By (8))} \\ = \; & 0. \end{aligned}$$

The result of (1) follows.

(2) and (3). Similar to the proof of (1). ■

Property 6. (1). If $g_2(b) < 0$, then for any optimal schedule σ, $a_{\sigma(K+1)} \leq a_{\sigma(K+2)} \leq \ldots \leq a_{\sigma(n)}$.

(2). If $g_2(b) \geq 0$, then there exists an optimal schedule σ such that $a_{\sigma(K+1)} \geq a_{\sigma(K+2)} \geq \ldots \geq a_{\sigma(n)}$.

Proof. (1). Assume that the schedule σ in which $a_{\sigma(i)} > a_{\sigma(i+1)}$ $(K+1 \leq i < n)$ is optimal. Let σ' be the schedule derived from σ by swapping i and $i+1$. Then,

$$\begin{aligned}
& f(C_{\sigma(K)}, \sigma) - f(C_{\sigma'(K)}, \sigma') \\
= \; & \beta(n+1-i)(a_{\sigma(i)} - bs_{\sigma(i)}) + \beta(n-i)(a_{\sigma(i+1)} - ba_{\sigma(i)} - b(1-b)s_{\sigma(i)}) \\
& -\beta(n+1-i)(a_{\sigma(i+1)} - bs_{\sigma(i)}) - \beta(n-i)(a_{\sigma(i)} - ba_{\sigma(i+1)} - b(1-b)s_{\sigma(i)}) \\
& -bm_{i+2}(a_{\sigma(i+1)} - a_{\sigma(i)}) \\
= \; & (\beta((n-i)b-1) - bm_{i+2})(a_{\sigma(i+1)} - a_{\sigma(i)}) \\
> \; & 0, \qquad \text{(By Property 5)}
\end{aligned}$$

a contradiction. The result of (1) follows.

(2). Assume that the schedule σ_1 in which $a_{\sigma_1(i)} < a_{\sigma_1(i+1)}$ $(K+1 \leq i < n)$, is optimal. Let σ_1' be the schedule derived from σ_1 by swapping i and $i+1$. Then,

$$\begin{aligned}
& f(C_{\sigma_1(K)}, \sigma_1) - f(C_{\sigma_1'(K)}, \sigma_1') \\
= \; & (\beta((n-i)b-1) - bm_{i+2})(a_{\sigma_1(i+1)} - a_{\sigma_1(i)}) \\
\geq \; & 0. \qquad \text{(By Property 5)}
\end{aligned}$$

So, σ_1' is an optimal schedule. Proceeding as above, we can get an optimal schedule σ such that

$$a_{\sigma(K+1)} \geq a_{\sigma(K+2)} \geq \ldots \geq a_{\sigma(n)}.$$

This completes the proof. ■

Property 7. (1). If $g_1(b) > 0$, then $h_1(i) = a(i-1) + n\gamma - m_{i+1}$ is an increasing function of i $(1 \leq i \leq K-1)$.

(2). If $g_1(b) < 0$, then $h_1(i) = \alpha(i-1) + n\gamma - m_{i+1}$ is a decreasing function of i $(1 \leq i \leq K-1)$.

(3). If $g_1(b) = 0$, then $h_1(1) = h_1(2) = \ldots = h_1(K-1)$.

Proof. (1). Since $m_{i+1} = (\alpha i + n\gamma)b + (1-b)m_{i+2}$ for $1 \leq i \leq K-1$, we have

$$\begin{aligned}
& h_1(i+1) - h_1(i) \\
= \; & \alpha i + n\gamma - m_{i+2} - \alpha(i-1) - n\gamma + m_{i+1} \\
= \; & \alpha + (m_{i+1} - m_{i+2}) \\
= \; & \alpha + (\alpha i + n\gamma)b + (1-b)m_{i+2} - m_{i+2} \\
= \; & \alpha + (\alpha i + n\gamma)b - bm_{i+2} \\
> \; & 0. \qquad\qquad \text{(By Property (3))}
\end{aligned}$$

This completes the proof of (1).

(2) and (3). Similar to the proof of (1). ■

Property 8. (1). If $g_2(b) > 0$, then $h_2(i) = (n+1-i)\beta - m_{i+1}$ is an increasing function of i $(K \leq i < n)$.

(2). If $g_2(b) < 0$, then $h_2(i) = (n+1-i)\beta - m_{i+1}$ is a decreasing function of i $(K \leq i < n)$.

(3). If $g_2(b) = 0$, then $h_2(K+1) = h_2(K+2) = \ldots = h_2(n)$.

Proof. (1). Since $m_{i+1} = b\beta(n-i) + (1-b)m_{i+2}$, for $K \leq i \leq n-2$, we have

$$\begin{aligned}
& h_2(i+1) - h_2(i) \\
= \; & (n-i)\beta - m_{i+2} - (n+1-i)\beta + m_{i+1}
\end{aligned}$$

$$
\begin{aligned}
&= -\beta + (m_{i+1} - m_{i+2}) \\
&= -\beta + b(n-i)\beta + (1-b)m_{i+2} - m_{i+2} \\
&= (b(n-i)-1)\beta - bm_{i+2} \\
&> 0. \qquad \text{(By Property 5)}
\end{aligned}
$$

This completes the proof of (1).

(2) and (3). Similar to the proof of (1). ■

The following theorem is easily seen from Property 4 and Property 6.

Theorem 9. (1). If $g_1(b) > 0$ and $g_2(b) < 0$, then for any optimal schedule σ, $a_{\sigma(1)} \geq a_{\sigma(2)} \geq \ldots \geq a_{\sigma(K)}$ and $a_{\sigma(K+1)} \leq a_{\sigma(K+2)} \leq \ldots \leq a_{\sigma(n)}$.

(2). If $g_1(b) > 0$ and $g_2(b) \geq 0$, then there exists an optimal schedule σ such that $a_{\sigma(1)} \geq a_{\sigma(2)} \geq \ldots \geq a_{\sigma(K)}$ and $a_{\sigma(K+1)} \geq a_{\sigma(K+2)} \geq \ldots \geq a_{\sigma(n)}$.

(3). If $g_1(b) \leq 0$ and $g_2(b) < 0$, then there exists an optimal schedule σ such that $a_{\sigma(1)} \leq a_{\sigma(2)} \leq \ldots \leq a_{\sigma(K)}$ and $a_{\sigma(K+1)} \leq a_{\sigma(K+2)} \leq \ldots \leq a_{\sigma(n)}$.

(4). If $g_1(b) \leq 0$ and $g_2(b) \geq 0$, then there exists an optimal schedule σ such that $a_{\sigma(1)} \leq a_{\sigma(2)} \leq \ldots \leq a_{\sigma(K)}$ and $a_{\sigma(K+1)} \geq a_{\sigma(K+2)} \geq \ldots \geq a_{\sigma(n)}$.

A polynomial-time algorithm

We first sort and re-label the n jobs so that they are in non-increasing order of their normal processing times, namely $a_1 \geq a_2 \geq \ldots \geq a_n$. In the following, we present an $O(n \log n)$ algorithm for $1|p_i(s_i) = a_i - bs_i| \sum(\alpha E_i + \beta T_i + \gamma d)$.

Algorithm A.

Step 1: Initialization.

$$i = 1, j = 1, k = n, S_1 = S_2 = \emptyset, m_{n+1} = 0,$$

$$\mathcal{J} = \{1, 2, \ldots, n\}, K = \lceil \tfrac{n\beta - n\gamma}{\alpha + \beta} \rceil.$$

Step 2: Compute the values of $m_{j+1}, m_{j+2}, m_{j+K+2}$, and

$$g_1(b) = \alpha + (\alpha + n\gamma)b - bm_{j+2},$$

$$g_2(b) \;=\; \beta((n-(K+1))b-1) - bm_{j+K+2}.$$

Step 3: If $K=1$ and $g_2(b) \geq 0$, go to Step 8;

If $K=1$ and $g_2(b) < 0$, go to Step 5;

If $K=n-1$ and $g_1(b) > 0$, go to Step 5;

If $K=n-1$ and $g_1(b) \leq 0$, go to Step 12.

Step 4: If $g_1(b) > 0$ and $g_2(b) \geq 0$, go to Step 8;

If $g_1(b) \leq 0$ and $g_2(b) \geq 0$, go to Step 12;

If $g_1(b) \leq 0$ and $g_2(b) < 0$, go to Step 16.

Step 5: Compute $L_{j,k} = (j-1)\alpha + n\gamma - m_{j+1} - (n+1-k)\beta + m_{k+1}$. If $L_{j,k} \geq 0$, set $k := k-1, m_{k+1} := b\beta(n-k) + (1-b)m_{k+1}, S_2 := S_2 \cup \{i\}$; otherwise, set: $j := j+1, m_{j+1} := \frac{m_{j+1} - (\alpha(j-1)+n\gamma)b}{1-b}$, $S_1 := S_1 \cup \{i\}$.

Step 6: $i := i+1$

Step 7: If $j = K+1$, set $S_2 := \mathcal{J} - S_1$. Let σ be the schedule obtained by arranging the jobs in non-increasing order of their normal processing times in S_1, followed by arranging the jobs in non-decreasing order of their normal processing times in S_2, and $d^* = C_{\sigma(K)}$. STOP.

If $k = K$, set $S_1 := \mathcal{J} - S_2$. Let σ be the schedule obtained by arranging the jobs in non-increasing order of their normal processing times in S_1, followed by arranging the jobs in non-decreasing order of their normal processing times in S_2, and $d^* = C_{\sigma(K)}$. STOP. Otherwise, go to Step 5.

Step 8: $k = K+1$.

Step 9: Compute $L_{j,k} = (j-1)\alpha + n\gamma - m_{j+1} - (n+1-k)\beta + m_{k+1}$. If $L_{j,k} \geq 0$, set $k := k+1, m_{k+1} := \frac{m_{k+1} - b(n+1-k)\beta}{1-b}, S_2 := S_2 \cup \{i\}$; otherwise, set: $j := j+1, m_{j+1} := \frac{m_{j+1} - (\alpha(j-1)+n\gamma)b}{1-b}$, $S_1 := S_1 \cup \{i\}$.

Step 10: $i := i+1$

Step 11: If $j = K+1$, set $S_2 := \mathcal{J} - S_1$. Let σ be the schedule obtained by

arranging the jobs in non-increasing order of their normal processing times in S_1, followed by arranging the jobs in non-increasing order of their normal processing times in S_2, and $d^* = C_{\sigma(K)}$. STOP.

If $k = n+1$, set $S_1 := \mathcal{J} - S_2$. Let σ be the schedule obtained by arranging the jobs in non-increasing order of their normal processing times in S_1, and followed by arranging the jobs in non-increasing order of their normal processing times in S_2, and $d^* = C_{\sigma(K)}$. STOP. Otherwise, go to Step 9.

Step 12: $i := n$.

Step 13: Compute $L_{j,k} = (j-1)\alpha + n\gamma - m_{j+1} - (n+1-k)\beta + m_{k+1}$. If $L_{j,k} \geq 0$, set $k := k-1, m_{k+1} := b\beta(n-k) + (1-b)m_{k+1}, S_2 := S_2 \cup \{i\}$; otherwise, set: $j := j+1, m_{j+1} := \frac{m_{j+1} - (\alpha(j-1)+n\gamma)b}{1-b}$, $S_1 := S_1 \cup \{i\}$.

Step 14: $i := i-1$

Step 15: If $j = K+1$, set $S_2 := \mathcal{J} - S_1$. Let σ be the schedule obtained by arranging the jobs in non-decreasing order of their normal processing times in S_1, followed by arranging the jobs in non-increasing order of their normal processing times in S_2, and $d^* = C_{\sigma(K)}$. STOP.

If $k = K$, set $S_1 := \mathcal{J} - S_2$. Let σ be the schedule obtained by arranging the jobs in non-decreasing order of their normal processing times in S_1, followed by arranging the jobs in non-increasing order of their normal processing times in S_2, and $d^* = C_{\sigma(K)}$. STOP. Otherwise, go to Step 13.

Step 16: $k := K+1, i := n$.

Step 17: Compute $L_{j,k} = (j-1)\alpha + n\gamma - m_{j+1} - (n+1-k)\beta + m_{k+1}$. If $K \geq 0$, set $k := k+1, m_{k+1} := \frac{m_{k+1} - b(n+1-k)\beta}{1-b}, S_2 := S_2 \cup \{i\}$; otherwise, set: $j := j+1, m_{j+1} := \frac{m_{j+1} - (\alpha(j-1)+n\gamma)b}{1-b}$, $S_1 := S_1 \cup \{i\}$.

Step 18: $i := i-1$

Step 19: If $j = K+1$, set $S_2 := \mathcal{J} - S_1$. Let σ be the schedule obtained by arranging the jobs in non-decreasing order of their normal processing times

in S_1, followed by arranging the jobs in non-decreasing order of their normal processing times in S_2, and $d^* = C_{\sigma(\lceil \frac{n}{2} \rceil)}$. STOP.

If $k = n + 1$, set $S_1 := \mathcal{J} - S_2$. Let σ be the schedule obtained by arranging the jobs in non-decreasing order of their normal processing times in S_1, followed by arranging the jobs in non-decreasing order of their normal processing times in S_2, $d^* = C_{\sigma(K)}$. STOP. Otherwise, go to Step 16.

To determine the computational complexity of Algorithm A, we note that Step 5, Step 9, Step 13 and Step 17 can be completed in $O(n)$ time, while Step 7, Step 11, Step 15 and Step 19 can be completed in $O(n \log n)$ time. Hence, the overall time complexity of the algorithm is $O(n \log n)$.

Property 10. Let σ be an optimal schedule and $L_{i,j} = (i-1)\alpha + n\gamma - m_{i+1} - (n+1-j)\beta + m_{j+1}$ $(1 \leq i \leq K, K+1 \leq j \leq n))$. If $L_{i,j} > 0$, then $a_{\sigma(i)} \leq a_{\sigma(j)}$. If $L_{i,j} < 0$, then $a_{\sigma(i)} \geq a_{\sigma(j)}$.

Proof. If $L_{i,j} > 0$, suppose to the contrary that $a_{\sigma(i)} > a_{\sigma(j)}$. Let σ' be the schedule derived from σ by swapping i and j. Then, $f(C_{\sigma(K)}, \sigma) - f(C_{\sigma'(K)}, \sigma') = L_{i,j}(a_{\sigma(i)} - a_{\sigma(j)}) > 0$. This is a contradiction. So, $a_{\sigma(i)} \leq a_{\sigma(j)}$. Similar to above, we have $a_{\sigma(i)} \geq a_{\sigma(j)}$ if $L_{i,j} < 0$. ■

For notational convenience, we define the following properties:

($\mathbf{P_1}$). If $L_{i,j} > 0$, then $a_{\sigma(i)} \leq a_{\sigma(j)}$. Moreover, if $L_{i,j} > 0$ and $a_{\sigma(i)} = a_{\sigma(j)}$, then $\sigma(j) < \sigma(i)$. If $L_{i,j} < 0$, then $a_{\sigma(i)} \geq a_{\sigma(j)}$. Moreover, if $L_{i,j} < 0$ and $a_{\sigma(i)} = a_{\sigma(j)}$, then $\sigma(i) < \sigma(j)$.

($\mathbf{P_2}$). $a_{\sigma(1)} \geq a_{\sigma(2)} \geq \ldots \geq a_{\sigma(K)}$ and $a_{\sigma(K+1)} \geq \ldots \geq a_{\sigma(n)}$. If $a_{\sigma(i)} = a_{\sigma(j)}$, then $\sigma(i) < \sigma(j)$ if $1 \leq i < j \leq K$, or $K+1 \leq i < j \leq n$.

($\mathbf{P_3}$). $a_{\sigma(1)} \geq a_{\sigma(2)} \geq \ldots \geq a_{\sigma(K)}$ and $a_{\sigma(K+1)} \leq \ldots \leq a_{\sigma(n)}$. If $a_{\sigma(i)} = a_{\sigma(j)}$, then $\sigma(i) < \sigma(j)$ if $1 \leq i < j \leq K$, and $\sigma(i) > \sigma(j)$ if $K+1 \leq i < j \leq n$.

($\mathbf{P_4}$). $a_{\sigma(1)} \leq a_{\sigma(2)} \leq \ldots \leq a_{\sigma(K)}$ and $a_{\sigma(K+1)} \leq \ldots \leq a_{\sigma(n)}$. If $a_{\sigma(i)} = a_{\sigma(j)}$, then $\sigma(i) > \sigma(j)$ if $1 \leq i < j \leq K$, or $K+1 \leq i < j \leq n$.

($\mathbf{P_5}$). $a_{\sigma(1)} \leq a_{\sigma(2)} \leq \ldots \leq a_{\sigma(K)}$ and $a_{\sigma(K+1)} \geq \ldots \geq a_{\sigma(n)}$. If $a_{\sigma(i)} = a_{\sigma(j)}$,

then $\sigma(i) > \sigma(j)$ if $1 \leq i < j \leq K$, and $\sigma(i) < \sigma(j)$ if $K+1 \leq i < j \leq n$.

($\mathbf{P_6}$). If $K_{i,j} = 0$, then $a_{\sigma(j)} \geq a_{\sigma(i)}$. Moreover, if $K_{i,j} = 0$ and $a_{\sigma(j)} = a_{\sigma(i)}$, then $\sigma(j) < \sigma(i)$.

($\mathbf{P_7}$). If $K_{i,j} = 0$, then $a_{\sigma(j)} \leq a_{\sigma(i)}$. Moreover, if $K_{i,j} = 0$ and $a_{\sigma(j)} = a_{\sigma(i)}$, then $\sigma(j) > \sigma(i)$.

The next property is easily seen from Algorithm A.

Property 11. Let σ be a schedule produced by Algorithm A.

(1) If $g_1(b) > 0$ and $g_2(b) \geq 0$, then σ satisfies properties $(P_1), (P_2)$ and (P_6).

(2) If $g_1(b) > 0$ and $g_2(b) < 0$, then σ satisfies properties $(P_1), (P_3)$ and (P_6).

(3) If $g_1(b) \leq 0$ and $g_2(b) < 0$, then σ satisfies properties $(P_1), (P_4)$ and (P_7).

(4) If $g_1(b) \leq 0$ and $g_2(b) \geq 0$, then σ satisfies properties $(P_1), (P_5)$ and (P_7).

Property 12. (1). If $g_1(b) > 0$ and $g_2(b) \geq 0$, then there exists an optimal schedule that satisfies properties $(P_1), (P_2), (P_6)$.

(2). If $g_1(b) > 0$ and $g_2(b) < 0$, then there exists an optimal schedule that satisfies properties $(P_1), (P_3), (P_6)$.

(3). If $g_1(b) \leq 0$ and $g_2(b) < 0$, then there exists an optimal schedule that satisfies properties $(P_1), (P_4), (P_7)$.

(4). If $g_1(b) \leq 0$ and $g_2(b) \geq 0$, then there exists an optimal schedule that satisfies properties $(P_1), (P_5), (P_7)$.

Proof. (1). Assume σ is an optimal schedule with properties $(P_1), (P_2)$, the result is easily obtained by Theorem 9 and Property 10. If there exist integers i, j with $1 \leq i \leq K, K+1 \leq j \leq n$ such that $L_{i,j} = 0$ and $a_{\sigma(i)} > a_{\sigma(j)}$ or $a_{\sigma(j)} = a_{\sigma(i)}$ and $\sigma(j) > \sigma(i)$, let σ_1 be a schedule derived from σ by swapping i and j. Then, $f(C_{\sigma(K)}, \sigma) - f(C_{\sigma_1(K)}, \sigma_1) = L_{i,j}(a_{\sigma(i)} - a_{\sigma(j)}) = 0$. So, σ_1 is an optimal schedule. By repeating the above procedure, we eventually obtain an optimal schedule σ' with properties $(P_1), (P_2)$ and (P_6).

(2), (3) and (4). Similar to the proof of (1). ■

Property 13. (1). If $g_1(b) > 0$ and $g_2(b) \geq 0$, let σ, σ' be two schedules with properties $(P_1), (P_2)$ and (P_6), then $\sigma(i) = \sigma'(i)$ for $1 \leq i \leq n$.

(2). If $g_1(b) > 0$ and $g_2(b) < 0$, let σ, σ' be two schedules with properties $(P_1), (P_3)$ and (P_6), then $\sigma(i) = \sigma'(i)$ for $1 \leq i \leq n$.

(3). If $g_1(b) \leq 0$ and $g_2(b) < 0$, let σ, σ' be two schedules with properties $(P_1), (P_4)$ and (P_7), then $\sigma(i) = \sigma'(i)$ for $1 \leq i \leq n$.

(4). If $g_1(b) \leq 0$ and $g_2(b) < 0$, let σ, σ' be two schedules with properties $(P_1), (P_5)$ and (P_7), then $\sigma(i) = \sigma'(i)$ for $1 \leq i \leq n$.

Proof. (1). It suffices to prove that $\sigma(i) = \sigma'(i)$ for $1 \leq i \leq K$. We proceed by induction on i. We distinguish two cases, depending on $g_2(b) > 0$ or $g_2(b) = 0$.

Case 1. $g_1(b) > 0$ and $g_2(b) > 0$.

If $i = 1$ and there exists an integer j_1 $(K + 1 \leq j_1 \leq n)$ such that $L_{1,j_1} = 0$. By Property 8, $L_{1,j} > 0$ if $K + 1 \leq j < j_1$. Then, $\sigma(1) = \sigma'(1) = j_1 - K + 1$. If $i = 1$, and there exists no integer j $(K + 1 \leq j \leq n)$ such that $L_{1,j} = 0$, then $\sigma(1) = \sigma'(1) = 1$ if $L_{1,K+1} < 0$. If $L_{1,K+1} > 0$, let j_1'=max$\{j | K_{1,j} > 0, K + 1 \leq j \leq n\}$, then $\sigma(1) = \sigma'(1) = j_1' - K + 1$.

Assume that for $i < k$, the result follows. We consider the case $i = k$. If there exists an integer j_k $(K + 1 \leq j_k \leq n)$ such that $L_{k,j_k} = 0$. By Property 7 and Property 8, $L_{k,j} < 0$ if $j > j_k$ and $L_{i,j_k} > 0$ if $i > k$. Then, $\sigma(k)$ =max$\{\sigma(k-1), \sigma(j_k - 1)\} + 2$. Similarly, $\sigma'(k)$ =max$\{\sigma'(k-1), \sigma'(j_k - 1)\} + 2$. By the induction hypothesis, $\sigma(i) = \sigma'(i)$ for $1 \leq i \leq k - 1$. This implies that $\sigma(j_k - 1) = \sigma'(j_k - 1)$. So, $\sigma(k) = \sigma'(k)$. If there exists no integer j $(K + 1 \leq j \leq n)$ such that $L_{k,j} = 0$, let $j_k = K + \sigma(k-1) - (k-1)$, then $\sigma(k) = \sigma'(k) = \sigma(k-1)+1$ if $L_{k,j_k+1} < 0$. If $L_{k,j_k+1} > 0$, let j_k' =max$\{j | L_{j,k} > 0\}$, then $\sigma(k) = \sigma'(k) = \sigma(k-1) + (j_k' - j_k) + 1$.

Case 2. $g_1(b) > 0$ and $g_2(b) = 0$.

If $L_{1,K+1} \geq 0$, then by Property 7, $L_{1,K+2} \geq 0, \ldots, L_{1,n} \geq 0$. So, $\sigma(1) = \sigma'(1) = (n-K)+1$. If $L_{1,K+1} < 0$, then $\sigma(1) = \sigma'(1) = 1$.

Assume that for $i < k$, the result follows. We consider the case $i = k$. If $L_{k,K+1} = 0$, then by Property 7 and Property 8, $L_{k-1,K+1} < 0, L_{k-1,K+2} < 0, \ldots, L_{k-1,j_k} < 0$, and $L_{k,j_k+1} = \ldots = L_{k,n} = 0$. So, $\sigma(k) = \sigma(k-1) + (n-K)+1$. Similarly, $\sigma'(k) = \sigma'(k-1) + (n-K)+1$. Then, $\sigma(k) = \sigma'(k)$. If $L_{k,K+1} < 0$, then $L_{1,K+1} < 0, L_{2,K+1} < 0, \ldots, L_{k-1,K+1} < 0$, so $\sigma(k) = \sigma'(k) = k$. If $L_{k,K+1} > 0$ and $L_{k-1,K+1} > 0$, then $\sigma(k) = \sigma(k-1)+1$. Similarly, $\sigma'(k) = \sigma'(k-1)+1$. By the induction hypothesis, $\sigma(k-1) = \sigma'(k-1)$. So $\sigma(k) = \sigma'(k)$. If $L_{k,K+1} > 0$ and $L_{k-1,K+1} < 0$, then $\sigma(k) = \sigma(k-1) + (n-K)+1$. Similarly, $\sigma'(k) = \sigma'(k-1) + (n-K)+1$. So, $\sigma(k) = \sigma'(k)$. This completes the proof. ■

Theorem 14. Algorithm A computes in time $O(n \log n)$ an optimal solution to the problem $1|p_i(s_i) = a_i - bs_i, d| \sum(\alpha E_i + \beta T_i + \gamma d)$.

Conclusions

This paper studies a single machine due-date scheduling problem of jobs with decreasing start-time dependent processing times. Our objective is to minimize the sum of earliness and tardiness. We show that the optimal schedule can be found in $O(n \log n)$ time. Future research may consider more general deterioration types.

Acknowledgments

This research was supported in part by The Hong Kong Polytechnic University under grant number G-YW81. The second author was also supported by the National Natural Science Foundation of China under grant number 10101010.

References

[1] Alidaee, B., 1991. Single machine scheduling with nonlinear cost functions. Computers and Operations Resarch 18(3), 317-322.

[2] Alidaee, B., Womer, N.K., 1999. Scheduling with time dependent processing times: review and extentions. Journal of the Operational Research Society 50, 711-720.

[3] Bachman, A., Cheng, T.C.E., Janiak, A., Ng, C.T., 2002. Scheduling start time dependent jobs to minimize the total weighted completion time. Journal of Operational Research Society 53, 688-693.

[4] Baker, K.R., Scudder, G.D., 1990. Sequencing with earliness and tardiness penalities: a review. Operations Research 38, 22-36.

[5] Brucker, P., 1995. Scheduling Algorithms. Springer, New York.

[6] Cheng, T.C.E., Ding, Q., 1998. The complexity of scheduling starting time dependent tasks with release date. Information Processing Letter 65, 75-79.

[7] Cheng, T.C.E., Ding, Q., Lin, B.M.T., 2004. A concise survey of scheduling with time-dependent processing times. European Journal of Operational Research 152, 1-13.

[8] Cheng, T.C.E., Chen, Z.L., Shakhlevich, N.V., 2002. Common due date assignment and scheduling with ready times. Computers and Operations Research 29, 1957-1967.

[9] Gordon, V., Porth, J.-M., Chu, C.B., 2002. A survey of the state-of-art of common due date assignment and scheduling research. European Journal of Operational Research 139, 1-25.

[10] Gordon, V.S., Porth, J.-M., Chu, C.B., 2002. Due date assignment and scheduling: SLK, TWK and other due date assignment models. Production Planning and Control 13, 117-132.

[11] Graham, R.L., Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., 1979. Optimization and approximation in deterministic sequencing and scheduling: a survey. Annals. of Discrete Mathematics 3, 287-326.

[12] Mosheio, G., 1991. V-shaped policies for scheduling deteriorating jobs. Operations Research 39(6), 979-991.

[13] Mosheiov, G., 1994. Scheduling jobs under simple linear deterioration. Computers and Operations Research 21(6), 653-659.

[14] Ng, C.T., Cheng, T.C.E., Bachman, A., Janiak, A., 2002. Three scheduling problems with deteriorating jobs to minimize the total completion time. Information Processing Letters 81, 327-333.