

Scheduling step-deteriorating jobs on a single machine with multiple critical dates

Byung-Cheon Choi, Eun-Seok Kim & Jun-Ho Lee

To cite this article: Byung-Cheon Choi, Eun-Seok Kim & Jun-Ho Lee (04 Nov 2024): Scheduling step-deteriorating jobs on a single machine with multiple critical dates, Journal of the Operational Research Society, DOI: [10.1080/01605682.2024.2419979](https://doi.org/10.1080/01605682.2024.2419979)

To link to this article: <https://doi.org/10.1080/01605682.2024.2419979>



Published online: 04 Nov 2024.



Submit your article to this journal [↗](#)



Article views: 144



View related articles [↗](#)



View Crossmark data [↗](#)

ORIGINAL ARTICLE



Scheduling step-deteriorating jobs on a single machine with multiple critical dates

Byung-Cheon Choi^a, Eun-Seok Kim^b and Jun-Ho Lee^a 

^aChungnam National University, Daejeon, Republic of Korea; ^bQueen Mary University of London, London, United Kingdom

ABSTRACT

This paper studies a scheduling problem with step-deteriorating jobs and multiple critical dates in a single machine environment, where each job's processing time increases stepwise at each critical date as its start time is delayed. Our goal is to minimize the total weighted completion time. First, we prove that the problem is NP-hard when there is one critical date and that even the unweighted case with two critical dates remains NP-hard and has inapproximability. We further establish the strong NP-hardness for the unweighted case with an arbitrary number of critical dates. As solution approaches, we develop a mixed integer linear programming formulation and develop a heuristic based on the weighted shortest processing time rule. Finally, extensive numerical experiments were designed according to the level of difficulty in evaluating the performance of the proposed heuristic.

ARTICLE HISTORY

Received 10 June 2024
Accepted 17 October 2024

KEYWORDS

scheduling; multiple critical dates; step-deteriorating; computational complexity

1. Introduction

Traditional scheduling models often assume that machines are continuously available without any interruptions. However, in real-world scenarios, production is frequently disrupted by various factors, including mechanical and electrical failures, unpredictable breakdowns, and scheduled maintenance activities (Bajestani et al., 2014; Pinedo, 2012). These interruptions inevitably lead to the deterioration of machine performance over time, resulting in a decline in production rates. Deterioration is a critical factor contributing to reduced production efficiency in many manufacturing sectors, such as thermoplastics (Ghaleb et al., 2020), semiconductors (Sloan & Shanthikumar, 2000), and systems that utilize cutting tools, cooling systems, and sensors (Bajestani et al., 2014). Consequently, jobs processed later on these machines typically experience longer processing times. This phenomenon can be effectively modeled by considering job processing times as functions of their starting times, referred to as *time-dependent* job processing times. Motivated by this, we examine single-machine scheduling for time-dependent jobs, where each job's processing time increases stepwise at predefined critical dates.

Our problem, referred to as *Problem P*, can be formally described as follows. Let $\mathcal{J} = \{1, 2, \dots, n\}$ and $\mathcal{D} = \{D_1, D_2, \dots, D_m\}$ denote the sets of jobs and critical dates, respectively. Each job $j \in \mathcal{J}$ has a weight w_j . The processing time of job $j \in \mathcal{J}$ is specified as a basic processing time a_j and a deteriorating ratio δ_i in $\{\delta_1, \delta_2, \dots, \delta_{m+1}\}$. If job j starts at t

and $D_{i-1} \leq t < D_i$ for $i \in \{1, 2, \dots, m+1\}$, then its processing time p_j is calculated as $\delta_i a_j$, where for consistency of notation, $D_0 = 0$ and $D_{m+1} = \infty$. To reflect the situation in which the performance of the machine becomes worse in proportion to the time of use, we assume that

$$1 \leq \delta_1 < \delta_2 < \dots < \delta_{m+1}. \quad (1)$$

Let these jobs be referred to as *step-deteriorating jobs*. A schedule σ is an allocation of the jobs in \mathcal{J} to a single machine, ensuring that each job receives its required processing time and that only one job can be processed on the machine at a time. For $j \in \mathcal{J}$, let $\sigma(j)$ denote the job in the j th position in σ , and let $S_j(\sigma)$ and $C_j(\sigma)$ be the start and the completion times of j in σ , respectively. Our goal is to find a schedule σ which minimizes the total weighted completion time (TWCT) of the jobs, denoted $z(\sigma) = \sum_{j \in \mathcal{J}} w_j C_j(\sigma)$. Note that by (1) and the regularity of the TWCT, there exists no idle time in an optimal schedule. Thus, we consider only the schedule with no idle time. To the best of our knowledge, *Problem P* has not been previously addressed in the scheduling literature with time-dependent job processing times.

Note that while the weighted shortest processing time (WSPT) rule provides an optimal solution with the TWCT measure in classical single-machine scheduling problems, it may not yield an optimal solution for our problem. Figure 1 shows a simple example in which the non-WSPT schedule is superior to the WSPT schedule from the TWCT

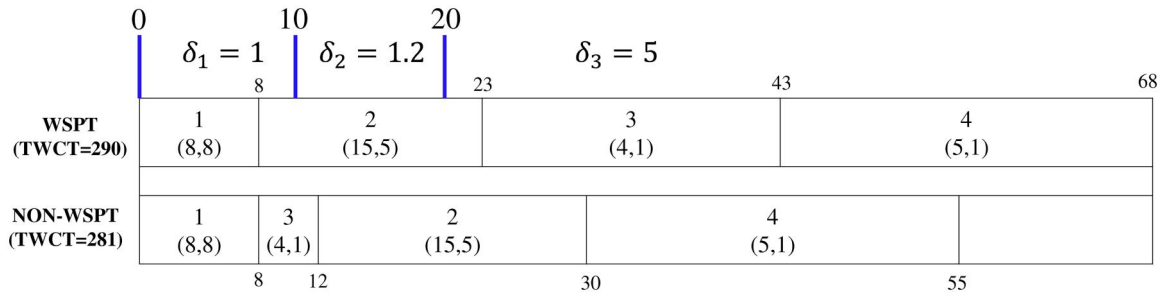


Figure 1. Example schedules.

standpoint. In Figure 1, each rectangle represents a job j with (a_j, w_j) on a single machine with $\{D_1, D_2\} = \{10, 20\}$ and $\{\delta_1, \delta_2, \delta_3\} = \{1, 1.2, 5\}$

Our contributions can be summarized as follows. First, we analyze computational complexity of the problem under consideration for the first time. Second, as the solution approaches, we present a mixed integer linear programming and an efficient heuristic. Finally, extensive numerical experiments designed according to the level of difficulty evaluate the performance of the proposed heuristic.

The remainder of this paper is structured as follows. Section 2 presents the literature review. Section 3 establishes the complexity results of our problem in various settings. Section 4 develops a mathematical model and a heuristic algorithm. Section 5 verifies the performance of the heuristic algorithm through the numerical experiments. Finally, Section 6 presents conclusions and future work.

2. Literature review

Gupta and Gupta (1988) and Browne and Yechiali (1990; Bajestani et al., 2014) first introduced deteriorating effects in single-machine makespan minimization problems. After that, extensive studies were conducted in various real scenarios (Cohen & Shapira, 2024; Luo et al., 2018; Lv et al., 2024; Lv & Wang, 2024; Ma et al., 2024; Mao et al., 2024; Wang et al., 2024; Zhang et al., 2024). Due to the vast amount of relevant studies, we focus on studies where the time-dependent job processing times follow the step-wise functions of their starting times. Refer to (Gawiejniewicz, 2020a; 2020b; Pei et al., 2023; Strusevich & Rustogi, 2017) for the comprehensive surveys. Sundararaghavan and Kunnathur (1994) considered a single-machine scheduling problem with a single common critical date D , in which the objective is to minimize the TWCT. The processing time of each job $j \in \mathcal{J}$ is calculated as

$$p_j = \begin{cases} a & \text{if } S_j(\sigma) \leq D \\ a + q_j & \text{if } S_j(\sigma) > D, \end{cases}$$

where q_j is the deteriorating value of job j started after D . They developed a heuristic algorithm that is

conjectured to obtain an optimal schedule. They also presented a 0-1 quadratic programming formulation and investigated some polynomially solvable cases. Note that the computational complexity remains open. Mosheiov (1995) considered a single- and multi-machine scheduling problem with the makespan criterion such that each job $j \in \mathcal{J}$ has its own set of critical dates, denoted $\mathcal{D}_j = \{D_j^1, D_j^2, \dots, D_j^m\}$. The processing time of each job $j \in \mathcal{J}$ is calculated as

$$p_j = \begin{cases} a_j^1 & \text{if } S_j(\sigma) \leq D_j^1 \\ a_j^k & \text{if } D_j^{k-1} < S_j(\sigma) \leq D_j^k \text{ for } k \in \{2, 3, \dots, m+1\}, \end{cases}$$

where $D_j^{m+1} = \infty$ for consistency of notation, and $a_j^k < a_j^{k+1}$ for $k \in \{1, 2, \dots, m\}$. They presented the integer programming formulations and heuristics for the single- and multi-machine cases with $|\mathcal{D}_j| = 1$ for $j \in \mathcal{J}$. Furthermore, they developed a heuristic for a single-machine case and conducted numerical experiments to evaluate its performance. Cheng and Ding (2001) considered single-machine scheduling problems with the job-dependent critical dates D_j for $j \in \mathcal{J}$. The processing time of each job $j \in \mathcal{J}$ is calculated as

$$p_j = \begin{cases} a_j & \text{if } S_j(\sigma) \leq D_j \\ a_j + q_j & \text{if } S_j(\sigma) > D_j. \end{cases}$$

They consider two objective criteria: the makespan and the TWCT. They showed that the problem of minimizing the makespan is NP-hard even for the case with $D_j = D$ for $j \in \mathcal{J}$. Furthermore, they investigated some conditions that make the problem of minimizing the TWCT polynomially solvable. Finally, they presented a counter-example that invalidates the optimality of a switching algorithm proposed in Sundararaghavan and Kunnathur (1994). Jeng and Lin (2004) showed that the problem with the makespan criterion in Cheng and Ding (2001) is weakly NP-hard by developing a pseudo-polynomial time dynamic programming algorithm. Then, they proposed dominance properties and a lower bound that help enhance the branch-and-bound algorithm. Cheng et al. (2020) showed that the problem with the total completion time criterion in Cheng and Ding (2001) is weakly NP-hard even for $w_j = 1$ for

$j \in \mathcal{J}$. A similar single-machine scheduling problem was considered in Cheng et al. (2006), Ji et al. (2007), Kim and Oron (2015) such that the processing time of each job $j \in \mathcal{J}$ is calculated as

$$p_j = \begin{cases} a_j & \text{if } S_j(\sigma) < D \\ a_j - b_j & \text{if } S_j(\sigma) \geq D, \end{cases}$$

where b_j is an improving value of job j started after D . For the makespan criterion, Cheng et al. (2006) proved the NP-hardness and proposed a pseudo-polynomial time algorithm and a fully polynomial time approximation algorithm from the standard results for the knapsack literature. Furthermore, Ji et al. (2007) developed a $\frac{5}{4}$ -approximation algorithm that can be executed in linear time. For the total completion time criterion, Kim and Oron (2015) proved the NP-hardness and investigated several polynomially solvable cases. They also developed a mixed integer programming formulation and a heuristic. Kim et al. (2022) considered the same problem in Kim and Oron (2015), except that the processing time of each job $j \in \mathcal{J}$ is calculated as

$$p_j = \begin{cases} a_j & \text{if } S_j(\sigma) < D \\ \delta a_j & \text{if } S_j(\sigma) \geq D, \end{cases}$$

where δ is an improving ratio of the job started after D and $0 < \delta < 1$. They proved the weak NP-hardness and developed a $(2 - \delta)$ -approximation algorithm.

3. Computational complexity

This section analyzes the computational complexity of *Problem P* in various settings. In particular, we show that *Problem P* is NP-hard even when there is one critical date ($|\mathcal{D}| = 1$) and that even the unweighted case with two critical dates ($|\mathcal{D}| = 2$) remains NP-hard and has inapproximability. We further establish the strong NP-hardness for the unweighted case with an arbitrary number of critical dates.

Theorem 1. *Problem P is NP-hard, even when $|\mathcal{D}| = 1$.*

Proof. We prove it by reduction from the following NP-complete problem (Garey & Johnson, 1979).

Equal cardinality partition problem (ECP):

Given $2b$ positive integers in $\{h_1, h_2, \dots, h_{2b}\}$ with $\sum_{j=1}^{2b} h_j = 2H$, is there a set \mathcal{H} satisfying

$$|\mathcal{H}| = b \text{ and } \sum_{j \in \mathcal{H}} h_j = H?$$

Without loss of generality, it is assumed that if $|\mathcal{H}| > b$, then

$$\sum_{j \in \mathcal{H}} h_j \geq H + 1. \quad (2)$$

For a given instance of the ECP, an instance of *Problem P*, called instance I , is constructed

as follows: There are $(2b + 2)$ jobs in $\mathcal{J} = \{1, 2, \dots, 2b + 2\}$ with

$$(w_j, a_j) = \begin{cases} (1, h_j) & \text{for } j \in \{1, 2, \dots, 2b\} \\ (M, M^3) & \text{for } j \in \{2b + 1, 2b + 2\}, \end{cases}$$

where $M > (2b + 1)H$. Let

$$D_1 = H + 1 \text{ and } (\delta_1, \delta_2) = (1, M).$$

We now show that there exists a solution \mathcal{H} to the ECP if and only if there exists a schedule σ for I with $z(\sigma) \leq K$, where

$$K = M^5 + 2M^4 + bM^3 + HM^2 + (b + 2)HM + 2bH.$$

Note that this reduction can be done in polynomial time.

(\Rightarrow) Suppose that there exists a solution $\bar{\mathcal{H}}$ to the ECP. Let $\bar{\mathcal{J}}_1$ denote the set of jobs with the basic processing times corresponding to the integers in $\bar{\mathcal{H}}$. Then, we construct a schedule

$$\bar{\sigma} = (\bar{\pi}_1, 2b + 1, \bar{\pi}_2, 2b + 2)$$

such that $\bar{\pi}_1$ and $\bar{\pi}_2$ are the sequences constructed by arbitrarily ordering the jobs in $\bar{\mathcal{J}}_1$ and $\bar{\mathcal{J}}_2 = \{1, 2, \dots, 2b\} \setminus \bar{\mathcal{J}}_1$, respectively. Then, it is observed that

$$(w_j, p_j) = \begin{cases} (1, h_j) & \text{for } j \in \bar{\mathcal{J}}_1 \\ (1, Mh_j) & \text{for } j \in \bar{\mathcal{J}}_2 \\ (M, M^3) & \text{for } j = 2b + 1 \\ (M, M^4) & \text{for } j = 2b + 2. \end{cases}$$

which implies that since

$$\sum_{j \in \bar{\mathcal{J}}_1} h_j = \sum_{j \in \bar{\mathcal{J}}_2} h_j = H \text{ and } \sum_{j \in \bar{\mathcal{J}}_1} w_j = \sum_{j \in \bar{\mathcal{J}}_2} w_j = b,$$

we have

$$\begin{aligned} z(\bar{\sigma}) &\leq \left(\sum_{j \in \bar{\mathcal{J}}_1} w_j \right) \sum_{j \in \bar{\mathcal{J}}_1} p_j + w_{2b+1} \sum_{j \in \bar{\mathcal{J}}_1 \cup \{2b+1\}} p_j \\ &+ \left(\sum_{j \in \bar{\mathcal{J}}_2} w_j \right) \sum_{j \in \bar{\mathcal{J}}_2 \cup \{2b+1\} \cup \bar{\mathcal{J}}_2} p_j + w_{2b+2} \sum_{j \in \bar{\mathcal{J}}} p_j = K. \end{aligned}$$

(\Leftarrow) Suppose that there exists a schedule $\hat{\sigma}$ with $z(\hat{\sigma}) \leq K$ for the instance I of *Problem P*. Without loss of generality, assume that in $\hat{\sigma}$

- Job $(2b + 1)$ is started before job $(2b + 2)$;
- The jobs started before (or after) D_1 are scheduled in the non-increasing order of w_j/a_j , respectively.

Claim 1. In $\hat{\sigma}$, at least one job in $\{2b + 1, 2b + 2\}$ must be started before D_1 .

Proof. Since $M^3 > D_1$, both jobs $(2b + 1)$ and $(2b + 2)$ cannot be started before D_1 . If both jobs $(2b + 1)$ and $(2b + 2)$ are started after D_1 , then

$p_{2b+1} = p_{2b+2} = M^4$, which implies that

$$z(\hat{\sigma}) > w_{2b+1}p_{2b+1} + w_{2b+2}p_{2b+2} = 2M^5 > K.$$

Therefore, at least one job in $\{2b+1, 2b+2\}$ must be started before D_1 . \square

Let $\hat{\mathcal{J}}_1$ be the set of the jobs before job $(2b+1)$, and $\hat{\mathcal{J}}_2 = \{1, 2, \dots, 2b\} \setminus \hat{\mathcal{J}}_1$. Note that since the jobs in $\hat{\mathcal{J}}_1$ and $\hat{\mathcal{J}}_2$ are started before and after D_1 , respectively, we have

$$(w_j, p_j) = \begin{cases} (1, h_j) & \text{for } j \in \hat{\mathcal{J}}_1 \\ (1, Mh_j) & \text{for } j \in \hat{\mathcal{J}}_2 \\ (M, M^3) & \text{for } j = 2b+1 \\ (M, M^4) & \text{for } j = 2b+2, \end{cases} \quad (3)$$

Claim 2. In $\hat{\sigma}$,

$$|\hat{\mathcal{J}}_1| = |\hat{\mathcal{J}}_2| = b.$$

Proof. Consider two cases. If $|\hat{\mathcal{J}}_1| < b$, then

$$|\hat{\mathcal{J}}_2| \geq b+1,$$

which implies that by (2, 3), and Claim 1, we have

$$\sum_{j \in \hat{\mathcal{J}}_2} p_j = \sum_{j \in \hat{\mathcal{J}}_2} Mh_j \geq (H+1)M \quad (4)$$

and

$$\sum_{j \in \hat{\mathcal{J}}_2} w_j C_j(\hat{\sigma}) \geq (b+1)p_{2b+1} = (b+1)M^3. \quad (5)$$

Then, by (3)–(5), we have

$$\begin{aligned} z(\hat{\sigma}) &> w_{2b+1}p_{2b+1} + \sum_{j \in \hat{\mathcal{J}}_2} w_j C_j(\hat{\sigma}) \\ &\quad + w_{2b+2} \left(p_{2b+1} + \sum_{j \in \hat{\mathcal{J}}_2} p_j + p_{2b+2} \right) \\ &\geq M^5 + 2M^4 + (b+1)M^3 + (H+1)M^2 > K. \end{aligned}$$

This is a contradiction. If $|\hat{\mathcal{J}}_1| > b$, then by (2) and (3), we have

$$\sum_{j \in \hat{\mathcal{J}}_1} p_j = \sum_{j \in \hat{\mathcal{J}}_1} h_j \geq H+1 = D_1,$$

which implies that job $(2b+1)$ cannot be started before D_1 . This is a contradiction to Claim 1. Thus, by two cases, Claim 2 holds. \square

Claim 3. In $\hat{\sigma}$,

$$\sum_{j \in \hat{\mathcal{J}}_1} h_j = H.$$

Proof. Consider two cases. If $\sum_{j \in \hat{\mathcal{J}}_1} h_j < H$, then

$$\sum_{j \in \hat{\mathcal{J}}_2} h_j \geq H+1 \quad (6)$$

and by (2, 3) and Claim 1,

$$\sum_{j \in \hat{\mathcal{J}}_2} w_j C_j(\hat{\sigma}) \geq bp_{2b+1} = bM^3. \quad (7)$$

Thus, by (3, 6) and (7), we have

$$\begin{aligned} z(\hat{\sigma}) &> w_{2b+1}p_{2b+1} + \sum_{j \in \hat{\mathcal{J}}_2} w_j C_j(\hat{\sigma}) \\ &\quad + w_{2b+2} \left(p_{2b+1} + \sum_{j \in \hat{\mathcal{J}}_2} p_j + p_{2b+2} \right) \\ &\geq M^5 + 2M^4 + bM^3 + (H+1)M^2 > K. \end{aligned}$$

This is a contradiction. If $\sum_{j \in \hat{\mathcal{J}}_1} h_j > H$, then job $(2b+1)$ cannot be started before D_1 . This is a contradiction to Claim 1. Thus, by two cases, Claim 3 holds. \square

Let $\hat{\mathcal{H}}$ be the set of the integers that correspond to the jobs in $\hat{\mathcal{J}}_1$. Then, by Claims 2 and 3, $\hat{\mathcal{H}}$ becomes the solution to the ECPP. \blacksquare

Theorem 2. Problem P is NP-hard, even when $|\mathcal{D}| = 2$ and $w_j = 1$ for $j \in \mathcal{J}$.

Proof. We prove it by reduction from the following NP-complete problem (Garey & Johnson, 1979).

Partition problem (PP): Given b positive integers in $\{h_1, h_2, \dots, h_b\}$ with $\sum_{j=1}^b h_j = 2H$, is there a set \mathcal{H} satisfying

$$\sum_{j \in \mathcal{H}} h_j = H?$$

Given an instance of the PP, an instance of Problem P, called instance I , is constructed as follows: There are $(b+2)$ jobs in $\mathcal{J} = \{1, 2, \dots, b+2\}$ with

$$a_j = \begin{cases} h_j & \text{for } j \in \{1, 2, \dots, b\} \\ M & \text{for } j = b+1 \\ M^2 & \text{for } j = b+2 \end{cases}$$

where $M > (b+2)(H+1) + 1$. Let

$$\begin{aligned} (D_1, D_2) &= (H+1, (H+1)(M+1)) \text{ and } (\delta_1, \delta_2, \delta_3) \\ &= (1, M, M^4). \end{aligned}$$

We now prove that there exists a solution $\bar{\mathcal{H}}$ to the PP if and only if there exists a schedule σ for I with $z(\sigma) \leq K$ where

$$K = (b+2)(H+1)(M+1) + M^3.$$

Note that this reduction can be done in polynomial time.

(\Rightarrow) Suppose that there exists a solution $\bar{\mathcal{H}}$ to the PP. Let $\bar{\mathcal{J}}$ be the set of jobs that correspond to the integers in $\bar{\mathcal{H}}$. Then, we construct a schedule

$$\bar{\sigma} = (\bar{\pi}_1, b+1, \bar{\pi}_2, b+2)$$

such that $\bar{\pi}_1$ and $\bar{\pi}_2$ are the sequences constructed by arbitrarily ordering the jobs in $\bar{\mathcal{J}}_1$ and $\bar{\mathcal{J}}_2 =$

$\{1, 2, \dots, b\} \setminus \bar{\mathcal{J}}_1$, respectively. Then, it is observed that

$$p_j = \begin{cases} h_j & \text{for } j \in \bar{\mathcal{J}}_1 \\ Mh_j & \text{for } j \in \bar{\mathcal{J}}_2 \\ M & \text{for } j = b+1 \\ M^3 & \text{for } j = b+2, \end{cases}$$

which implies since

$$\sum_{j \in \bar{\mathcal{J}}_1} h_j = \sum_{j \in \bar{\mathcal{J}}_2} h_j = H,$$

we have

$$z(\hat{\sigma}) \leq (b+1) \sum_{j=1}^{b+1} p_j + \sum_{j=1}^{b+2} p_j = K - (b+2) < K.$$

(\Leftarrow) Suppose that there exists a schedule $\hat{\sigma}$ with $z(\hat{\sigma}) \leq K$ for I of Problem P

Claim 1. In $\hat{\sigma}$,

$$S_j(\hat{\sigma}) < D_2 \text{ for } j \in \mathcal{J}.$$

Proof. Suppose that Claim 1 does not hold in $\hat{\sigma}$. Let l be the job that is started after D_2 . Then, by $p_l \geq M^4$, we have

$$z(\hat{\sigma}) > p_l \geq M^4 > K.$$

This is a contradiction. \square

Claim 2. In $\hat{\sigma}$,

$$S_{b+1}(\hat{\sigma}) < D_1.$$

Proof. Suppose that Claim 2 does not hold. Then, since $p_{b+1} \geq M^2 > D_2$ and $p_{b+2} \geq M^2 > D_2$, one of jobs in $\{b+1, b+2\}$ should be started after D_2 in $\hat{\sigma}$. This is a contradiction to Claim 1. \square

Let $\hat{\mathcal{J}}_1$ be the set of the jobs before job $(b+1)$, and $\hat{\mathcal{J}}_2 = \{1, 2, \dots, b\} \setminus \hat{\mathcal{J}}_1$

Claim 3. In $\hat{\sigma}$,

$$\sum_{j \in \hat{\mathcal{J}}_1} h_j = H.$$

Proof. Consider two cases. If $\sum_{j \in \hat{\mathcal{J}}_1} h_j < H$, then

$$\sum_{j \in \hat{\mathcal{J}}_2} h_j \geq H+1,$$

which implies that

$$\begin{aligned} S_{\hat{\sigma}(b+2)}(\hat{\sigma}) &= \sum_{j \in \hat{\mathcal{J}}_1} p_j + p_{b+1} + \sum_{j \in \hat{\mathcal{J}}_2} p_j > (H+2)M \\ &> (H+1)(M+1) = D_2. \end{aligned}$$

This is a contradiction to Claim 1. If $\sum_{j \in \hat{\mathcal{J}}_1} h_j > H$, then

$$\sum_{j \in \hat{\mathcal{J}}_1} h_j \geq H+1 = D_1,$$

which implies that job $(b+1)$ is started after D_1 . This is a contradiction to Claim 2. By two cases, Claim 3 holds. \square

Let $\hat{\mathcal{H}}$ be the set of the integers that correspond to the jobs in $\hat{\mathcal{J}}_1$. Then, by Claim 3, a set $\hat{\mathcal{H}}$ becomes the solution to the PP. \blacksquare

Theorem 3. Unless $P = NP$, no c -approximation algorithm exists even for Problem P with $|\mathcal{D}| = 2$ and $w_j = 1$ for $j \in \mathcal{J}$ for any fixed value $c > 1$.

Proof. Suppose that there exists a c -approximation algorithm. Consider the instance identical to the one in the proof of Theorem 2 except

$$K = c((b+2)(H+1)(M+1) + M^3).$$

It is observed that

$$z(\sigma) \leq (b+2)(H+1)(M+1) + M^3 \text{ or } z(\sigma) \geq M^4 > K. \quad (8)$$

Let σ^A be a schedule obtained by c -approximation algorithm, and σ^* be an optimal schedule. Consider two cases.

i. $z(\sigma^A) \leq K$

By (8), we have

$$z(\sigma^A) \leq (b+2)(H+1)(M+1) + M^3,$$

which implies that Claim 3 in the proof of Theorem 2 holds in σ^A . Thus, in this case, the PP has a solution;

ii. $z(\sigma^A) > K$

Suppose that $z(\sigma^*) \leq K$. By (8), we have

$$z(\sigma^*) \leq (b+2)(H+1)(M+1) + M^3.$$

Then, since $z(\sigma^A) \leq c \cdot z(\sigma^*)$, we have

$$K < z(\sigma^A) \leq c \cdot z(\sigma^*) \leq K.$$

This is a contradiction. Thus,

$$z(\sigma^*) > K,$$

which implies that the given PP has no solution.

Two cases imply that by using the c -approximation algorithm, we can check whether the PP has a solution in polynomial time. This is a contradiction unless $P = NP$. \blacksquare

Theorem 4. Problem P is strongly NP-hard, even if $w_j = 1$ for $j \in \mathcal{J}$.

Proof. We prove it by reduction from the following NP-complete problem (Garey & Johnson, 1979).

3-partition problem (3-PP): Given $3b$ positive integers in $\{h_1, h_2, \dots, h_{3b}\}$ with

$$\sum_{j=1}^{3b} h_j = bH \text{ and } \frac{H}{4} < h_j < \frac{H}{2} \text{ for } j \in \{1, 2, \dots, 3b\}, \quad (9)$$

is there a partition $(\mathcal{H}_l)_{l=1,2,\dots,b}$ satisfying

$$\bigcup_{l=1}^b \mathcal{H}_l = \{1, 2, \dots, 3b\} \text{ and } \mathcal{H}_i \cap \mathcal{H}_j = \emptyset \text{ for } i \neq j$$

and

$$\sum_{j \in \mathcal{H}_1} h_j = \sum_{j \in \mathcal{H}_2} h_j = \dots = \sum_{j \in \mathcal{H}_b} h_j = H?$$

Given an instance of the 3-PP, we can construct an instance I of *Problem P* as follows: Let

$$a_j = \begin{cases} h_j & \text{for } j \in \mathcal{J}_S = \{1, 2, \dots, 3b\} \\ M & \text{for } j \in \mathcal{J}_L = \{3b+1, 3b+2, \dots, 4b\}, \end{cases}$$

where M is a sufficiently large number. Let

$$D_l = \frac{1}{2}l(l+1)H + \frac{1}{2}l(l-1)M + 1 \text{ for } l \in \{1, 2, \dots, b\},$$

and

$$\delta_l = \begin{cases} l & \text{for } l \in \{1, 2, \dots, b\} \\ M & \text{for } l = b+1. \end{cases}$$

We now show that there exists a solution $(\mathcal{H}_l)_{l=1,2,\dots,b}$ to the 3-PP if and only if there exists a schedule σ for I with $z(\sigma) \leq K$ where

$$K = \frac{1}{6}b(b+1)((4b-1)M + (4b+10)H).$$

Note that this reduction can be done in polynomial time.

(\Rightarrow) Suppose that there exists a solution $(\mathcal{H}_l)_{l=1,2,\dots,b}$ to the 3-partition problem. Let $\bar{\mathcal{J}}_l$ denote the set of jobs that correspond to the integers in \mathcal{H}_l for $l \in \{1, 2, \dots, b\}$. Note that

$$\sum_{j \in \bar{\mathcal{J}}_l} h_j = H \text{ for } l \in \{1, 2, \dots, b\}. \quad (10)$$

Then, we construct a schedule

$$\bar{\sigma} = (\bar{\pi}_1, 3b+1, \bar{\pi}_2, 3b+2, \dots, \bar{\pi}_u, 4b),$$

such that $\bar{\pi}_l$ is the sequence constructed by arbitrarily ordering the jobs in $\bar{\mathcal{J}}_l$ for $l \in \{1, 2, \dots, b\}$. Then, it is observed that

- By (10), we have

$$\begin{aligned} S_{3b+k}(\bar{\sigma}) &= \sum_{l=1}^k (l \sum_{j \in \bar{\mathcal{J}}_l} h_j + (l-1)M) \\ &= \frac{1}{2}k(k+1)H + \frac{1}{2}k(k-1)M < D_k, \end{aligned}$$

which implies that

$$p_{3b+k} = kM \text{ for each } k \in \{1, 2, \dots, b\};$$

- Since the jobs in $\bar{\mathcal{J}}_k$ are started after D_{k-1} and completed before D_k ,

$$p_j = kh_j \text{ for each } j \in \bar{\mathcal{J}}_k \text{ and } k \in \{1, 2, \dots, b\};$$

By these observations, for $k \in \{1, 2, \dots, b\}$,

$$\begin{aligned} \sum_{j \in \bar{\mathcal{J}}_k} C_j(\bar{\sigma}) &< 3 \sum_{l=1}^k (l \sum_{j \in \bar{\mathcal{J}}_l} h_j + (l-1)M) \\ &= \frac{3}{2}k((k+1)H + (k-1)M) \end{aligned}$$

and

$$C_{3b+k}(\bar{\sigma}) = \sum_{l=1}^k (l \sum_{j \in \bar{\mathcal{J}}_l} h_j + lM) = \frac{1}{2}k(k+1)(H+M),$$

which implies that

$$\begin{aligned} z(\bar{\sigma}) &< \sum_{k=1}^b \frac{3}{2}k((k+1)H + (k-1)M) \\ &\quad + \sum_{k=1}^b \frac{1}{2}k(k+1)(H+M). \end{aligned}$$

From this, it directly follows that

$$z(\bar{\sigma}) < \frac{1}{6}b(b+1)((4b-1)M + (4b+10)H) = K.$$

(\Leftarrow) Suppose that there exists a schedule $\hat{\sigma}$ with $z(\hat{\sigma}) \leq K$ for I . Without loss of generality, the jobs in \mathcal{J}_L are assumed to be scheduled in the increasing order of indexes under $\hat{\sigma}$

Claim 1. In $\hat{\sigma}$,

$$D_{l-1} \leq S_{3b+l}(\hat{\sigma}) < D_l \text{ for } l \in \{1, 2, \dots, b\}.$$

Proof. Suppose that Claim 1 does not hold. Then, since $D_l - D_{l-1} < lM$, at most one job in \mathcal{J}_L can be started within the interval $[D_{l-1}, D_l]$ for $l \in \{1, 2, \dots, b\}$. Thus, at least one job in \mathcal{J}_L is started after D_b in $\hat{\sigma}$, which implies that

$$z(\hat{\sigma}) > C_{4b}(\hat{\sigma}) > M^2 > K.$$

This is a contradiction. \square

Let $\hat{\mathcal{J}}_1$ be the set of the jobs before job $(3b+1)$, and $\hat{\mathcal{J}}_l$ be the set of the jobs between jobs $(3b+l-1)$ and $(3b+l)$ for $l \in \{2, 3, \dots, b\}$

Claim 2. In $\hat{\sigma}$,

$$\sum_{j \in \hat{\mathcal{J}}_l} h_j = H \text{ for } l \in \{1, 2, \dots, b\}.$$

Proof. First, we show that

$$\sum_{j \in \bigcup_{l=1}^k \hat{\mathcal{J}}_l} h_j \leq kH \text{ for } k \in \{1, 2, \dots, b\}. \quad (11)$$

Suppose that k' is the smallest index with

$$\sum_{j \in \bigcup_{l=1}^{k'} \hat{\mathcal{J}}_l} h_j > k'H,$$

which implies that

$$\sum_{j \in \bigcup_{l=k}^{k'} \hat{\mathcal{J}}_l} h_j > (k' - k + 1)H \text{ for } k \in \{1, 2, \dots, k'\}. \quad (12)$$

Then, by (12), we have

$$\begin{aligned} \sum_{k=1}^{k'} k \sum_{j \in \hat{\mathcal{J}}_k} h_j &= \sum_{k=1}^{k'} \sum_{j \in \bigcup_{l=k}^{k'} \hat{\mathcal{J}}_l} h_j \geq \sum_{k=1}^{k'} (k' - k + 1)H + 1 \\ &= \sum_{k=1}^{k'} kH + 1, \end{aligned} \quad (13)$$

which implies that

$$\begin{aligned} \sum_{k=1}^{k'} k \sum_{j \in \hat{\mathcal{J}}_k} h_j + \sum_{k=1}^{k'-1} kM &\geq \frac{1}{2} k'(k' + 1)H \\ &+ \frac{1}{2} k'(k' - 1)M + 1 = D_{k'}. \end{aligned}$$

This implies that job $(3b + k')$ cannot be started before $D_{k'}$. This is a contradiction to Claim 1. Thus, (11) should be satisfied. Let k_1 be the smallest index with

$$\sum_{j \in \bigcup_{l=1}^{k_1} \hat{\mathcal{J}}_l} h_j < k_1 H. \quad (14)$$

If k_1 does not exist, then Claim 2 immediately holds. To exclude this case, we assume that an index k_1 exists. Furthermore, by (11) and (14), we have

$$\sum_{j \in \hat{\mathcal{J}}_k} h_j = H \text{ for } k \in \{1, 2, \dots, k_1 - 1\}. \quad (15)$$

Let k_2 be the smallest index with $k_1 < k_2$ and

$$\sum_{j \in \bigcup_{l=k_1}^{k_2} \hat{\mathcal{J}}_l} h_j = (k_2 - k_1 + 1)H. \quad (16)$$

Then, by (14)–(16), we have

$$\sum_{j \in \hat{\mathcal{J}}_{k_2}} h_j > H$$

and

$$\sum_{j \in \bigcup_{l=k}^{k_2} \hat{\mathcal{J}}_l} h_j \geq (k_2 - k + 1)H \text{ for } k \in \{1, 2, \dots, k_2 - 1\},$$

which implies that

$$\begin{aligned} \sum_{k=1}^{k_2} k \sum_{j \in \hat{\mathcal{J}}_k} h_j &= \sum_{k=1}^{k_2} \sum_{j \in \bigcup_{l=k}^{k_2} \hat{\mathcal{J}}_l} h_j \geq \sum_{k=1}^{k_2} (k_2 - k + 1)H + 1 = \sum_{k=1}^{k_2} kH + 1. \end{aligned} \quad (17)$$

Thus, by (17), we have

$$\begin{aligned} \sum_{k=1}^{k_2} k \sum_{j \in \hat{\mathcal{J}}_k} h_j + \sum_{k=1}^{k_2-1} kM &\geq \frac{1}{2} k_2(k_2 + 1)H \\ &+ \frac{1}{2} k_2(k_2 - 1)M + 1 = D_{k_2}, \end{aligned}$$

which implies that job $(3b + k_2)$ cannot be started before D_{k_2} . This is a contradiction to Claim 1. Thus,

$$\sum_{j \in \bigcup_{l=1}^k \hat{\mathcal{J}}_l} h_j = kH \text{ for } k \in \{1, 2, \dots, b\}. \quad (18)$$

Then, by (9) and (18), Claim 2 holds. \square

By Claim 2, it is observed that

$$\sum_{j \in \hat{\mathcal{J}}_l} h_j = H \text{ for } l \in \{1, 2, \dots, b\}.$$

Note that by (9), $|\hat{\mathcal{J}}_l| = 3$ for $l \in \{1, 2, \dots, b\}$. Let $\hat{\mathcal{H}}_l$ be the set of the integers corresponding to the jobs in $\hat{\mathcal{J}}_l$ for $l \in \{1, 2, \dots, b\}$. Then, a partition $(\hat{\mathcal{H}}_l)_{l=1,2,\dots,b}$ becomes the solution to the 3-PP. \blacksquare

4. Mathematical modeling & heuristic Algorithm

In this section, we first present a mixed integer linear programming (MILP) formulation for *Problem P* and then propose a heuristic. For simplicity, the jobs are re-indexed such that

$$\frac{w_1}{a_1} \geq \frac{w_2}{a_2} \geq \dots \geq \frac{w_n}{a_n}. \quad (19)$$

4.1. MILP

In order to develop a MILP formulation for *Problem P*, we introduce an additional notation

$$\mathcal{I} := \{1, 2, \dots, m + 1\}.$$

The main idea of the formulation is to find an optimal assignment of jobs to multiple periods. Once the assignment of the jobs to periods is obtained, an optimal schedule for the given assignment can be found by sequencing jobs based on the WSPT rule in each period. The decision variables are defined as follows: For $i \in \mathcal{I}$ and $j \in \mathcal{J}$, let S_j and C_j be the start and the completion times of job j , respectively, and

$$x_{i,j} = \begin{cases} 1 & \text{if } D_{i-1} \leq S_j < D_i \\ 0 & \text{otherwise.} \end{cases}$$

MILP

$$\text{minimize } \sum_{j \in \mathcal{J}} w_j C_j \quad (20)$$

subject to

$$\sum_{i \in \mathcal{I}} x_{i,j} = 1 \text{ for } j \in \mathcal{J} \quad (21)$$

$$S_j \geq D_{i-1} x_{i,j} \text{ for } i \in \mathcal{I} \text{ and } j \in \mathcal{J} \quad (22)$$

$$S_j + M(x_{i,j} - 1) < D_i \text{ for } i \in \mathcal{I} \text{ and } j \in \mathcal{J} \quad (23)$$

$$S_j - (S_k + \delta_i a_k) \geq M(x_{i,j} + x_{i,k} - 2) \text{ for } i \in \mathcal{I} \text{ and } \{j, k | 1 \leq k < j \leq n\} \subseteq \mathcal{J} \quad (24)$$

$$S_j - (S_k + \delta_i a_k) \geq M \left(\sum_{p=i+1}^{m+1} x_{p,j} + x_{i,k} - 2 \right) \text{ for } i \in \mathcal{I} \text{ and } \{j, k\} \subseteq \mathcal{J} \quad (25)$$

$$C_j = S_j + \sum_{i \in \mathcal{I}} \delta_i a_j x_{i,j} \text{ for } j \in \mathcal{J} \quad (26)$$

$$S_j \geq 0 \text{ and } C_j \geq 0 \text{ for } j \in \mathcal{J}, \\ x_{i,j} \in \{0, 1\} \text{ for } i \in \mathcal{I} \text{ and } j \in \mathcal{J},$$

where M is a sufficiently large number. The objective is to minimize the TWCT. By constraint (21), a job should start in a single period. If $x_{i,j} = 1$, then by constraints (22) and (23), we have

$$D_{i-1} \leq S_j < D_i.$$

By constraint (24), the jobs assigned in the same period should be sequenced in a non-decreasing order of job indexes. If $p > i$, then by constraint (25), each job assigned to period p should start its processing after the completion of a job in period i . Finally, constraint (26) represents the relation between S_j and C_j .

4.2. Heuristic algorithm

In this section, we propose a heuristic algorithm. The main idea of the algorithm is to construct the initial schedule using the WSPT rule and improve it by rescheduling the jobs that started before and are completed after the critical date. For simplicity, let σ^0 be the schedule constructed by the WSPT rule. The proposed heuristic algorithm can be formally described as follows:

Algorithm H

- Step 1 Set $i \leftarrow 1$, $j \leftarrow 1$, $C \leftarrow 0$ and $\sigma \leftarrow \sigma^0$.
 Step 2 If $j = n$, then return σ and STOP.
 Step 3 If $C < D_i < C + a_{\sigma(j)} \delta_i$, then construct a new schedule σ' in which jobs $\sigma(j)$ and $\sigma(j+1)$ are exchanged; otherwise, go to Step 7.
 Step 4 If $z(\sigma') < z(\sigma)$, then set $\sigma \leftarrow \sigma'$; otherwise, go to Step 6.
 Step 5 If $C + a_{\sigma(j)} \delta_i < D_i$, then set $j \leftarrow j + 1$, $C \leftarrow C + a_{\sigma(j)} \delta_i$, and go to Step 2.
 Step 6 If $i \leq m$, then set $i \leftarrow i + 1$.
 Step 7 Set $j \leftarrow j + 1$, $C \leftarrow C + a_{\sigma(j)} \delta_i$, and go to Step 2.

The time complexity of Algorithm H is $O(\max\{nm, n \log n\})$ because i and j are bounded by $O(n)$ and $O(m)$, respectively, and σ^0 can be obtained in $O(n \log n)$. The performance of the proposed algorithm, Algorithm H, is numerically tested in the following section.

5. Numerical experiments

In this section, we conduct numerical experiments to verify Algorithm H's performance by comparing it with the MILP and WSPT rule. For the extensive numerical experiments, we introduce several parameters as follows:

- $\alpha > 1$ is the parameter for the periods between two consecutive critical dates whose lengths are randomly determined in $[\frac{1}{2m} \sum a_j, \frac{\alpha}{2m} \sum a_j]$;
- $\beta > 1$ is the parameter for the deteriorating rates of periods that are randomly generated in $[1, \beta]$ and then assigned to $\delta_1, \delta_2, \dots, \delta_{m+1}$ in non-decreasing order of the generated values;
- $\gamma < 1$ is the positive parameter for a_j that is randomly chosen in $[30 \times (1 - \gamma), 30 \times (1 + \gamma)]$;
- $\zeta > 1$ is the parameter for the weights that are randomly generated in $[1, \zeta]$.

It is worth noting that if α is extremely large, a sufficiently long period can be generated and then all jobs might be processed in a single period, which can be solved by the WSPT rule. Furthermore, if $\beta = 1 + \epsilon$ or $\gamma = \epsilon$ where $\epsilon > 0$ is sufficiently small, then Problem P can be solved by the WSPT rule, and if $\zeta = 1 + \epsilon$, then Problem P becomes the unweighted version. Thus, it might be argued that Problem P tends to be more difficult with a decrease in the value of α and an increase in the values of β , γ and ζ . Based on this argument, we design our numerical tests with three scenarios that have different difficulty levels, as shown in Table 1

Table 2 shows the experimental results with m of 1; that is, there is a single critical date. We considered three scenarios shown in Table 1, and $n \in \{10, 15, 20, 25, 30\}$. For each combination of the scenario and n (i.e., 15 combinations in total), we randomly generated 20 instances, each of which was solved by the MILP, the WSPT rule, and Algorithm H. For the MILP, Gurobi(v11.0.1) was used on a PC(CPU: Intel Core i7-9700, RAM: 32GB), and the computation time limit was set to 1 h(= 3600s). If an instance is not solved to optimality within the

Table 1. Scenarios with different levels of difficulty.

Scenarios	Parameter setting
(1) Low difficulty	$\alpha = 5$, $\beta = 1.5$, $\gamma = 0.1$, $\zeta = 1.5$
(2) Medium difficulty	$\alpha = 3$, $\beta = 3$, $\gamma = 0.5$, $\zeta = 3$
(3) High difficulty	$\alpha = 1$, $\beta = 4.5$, $\gamma = 0.9$, $\zeta = 4.5$

Table 2. Numerical results ($m = 1$).

$m = 1$						
Scenario	n	Gap(%) MILP vs H	Gap(%) H vs WSPT	Optimality Ratio(%)	MILP Gap(%)	MILP CT(s)
(1)	10	0.00	0.00	100	0.00	0.04
	15	0.00	0.00	100	0.00	0.37
	20	0.00	0.00	100	0.00	3.13
	25	0.00	0.00	100	0.00	146.44
	30	0.01	0.02	75	5.62	1121.05
(2)	10	0.15	0.62	100	0.00	0.06
	15	0.10	0.72	100	0.00	0.41
	20	0.12	0.47	100	0.00	7.48
	25	0.05	0.39	100	0.00	348.12
	30	0.11	0.24	40	12.12	2296.95
(3)	10	0.03	2.15	100	0.00	0.09
	15	1.01	1.18	100	0.00	0.36
	20	0.71	1.84	100	0.00	5.75
	25	0.57	1.15	100	0.00	114.98
	30	0.53	2.12	90	0.87	1537.19

time limit, then the current best solution was used and the computation time is recorded as 3600s. Algorithm H and the WSPT rule were coded in Java. In Table 2, the third and fourth columns indicate the average gaps in the objective values between solutions by the MILP and Algorithm H, and Algorithm H and the WSPT rule, respectively. Specifically, the gaps are computed as follows:

$$\text{Gap}(\%) \text{ (MILP vs Algorithm H)} = \frac{z(\sigma^H) - z(\sigma^M)}{z(\sigma^M)} \times 100(\%) \quad (27)$$

$$\text{Gap}(\%) \text{ (Algorithm H vs WSPT)} = \frac{z(\sigma^W) - z(\sigma^H)}{z(\sigma^H)} \times 100(\%) \quad (28)$$

where σ^H , σ^W , and σ^M represent the schedule obtained by Algorithm H, the WSPT rule and the MILP, respectively. The fifth column of Table 2 shows the ratio of instances solved to optimality within one hour out of 20 instances. The sixth column represents the average MILP gap(%); the MILP gap is 0 if the problem is solved to optimality, and otherwise, it represents the gap between the current best objective value and the lower bound found within the computation time limit. The last column indicates the average CPU time of the MILP. Note that computation times of the WSPT rule and Algorithm H are negligible, less than 0.1 s, and hence they are not reported in the table.

As shown in Table 2, when $m = 1$, most instances are solved to optimality by the MILP within the computation time limit. The gaps between the solutions by the MILP and Algorithm H are very small; in most cases, the average gaps are less than 1%. For Scenarios (1) and (2), the WSPT rule performs well even though it is outperformed by Algorithm H. For Scenario (3), the performance of the WSPT rule is degraded, whereas Algorithm H still provides near-optimal solutions. This trend intensifies as m becomes larger, as shown in Tables 3 and 4, which

Table 3. Numerical results ($m = 2$).

$m = 2$						
Scenario	n	Gap(%) MILP vs H	Gap(%) H vs WSPT	Optimality Ratio(%)	MILP Gap(%)	MILP CT(s)
(1)	10	0.00	0.07	100	0.00	1.11
	15	0.01	0.02	100	0.00	31.83
	20	0.01	0.03	65	6.62	1663.85
	25	0.06	0.07	35	21.20	2497.28
	30	0.04	0.04	5	36.36	3469.58
(2)	10	0.26	1.23	100	0.00	0.62
	15	0.63	1.15	100	0.00	37.86
	20	0.39	1.31	40	12.20	2683.26
	25	0.24	0.79	5	37.41	3464.81
	30	0.35	0.64	0	50.97	3600.00
(3)	10	2.56	2.76	100	0.00	0.34
	15	1.01	2.43	100	0.00	8.26
	20	1.93	2.66	100	0.00	316.01
	25	0.58	1.04	30	21.18	3084.39
	30	1.08	2.56	0	60.54	3600.00

Table 4. Numerical results ($m = 3$).

$m = 3$						
Scenario	n	Gap(%) MILP vs H	Gap(%) H vs WSPT	Optimality Ratio(%)	MILP Gap(%)	MILP CT(s)
(1)	10	0.02	0.00	100	0.00	10.44
	15	0.02	0.04	65	6.48	1607.01
	20	0.03	0.08	20	19.64	3148.53
	25	0.02	0.02	0	40.43	3600.00
	30	0.03	0.07	0	49.20	3600.00
(2)	10	0.51	1.81	100	0.00	11.51
	15	0.57	1.46	90	1.20	1672.48
	20	0.47	1.10	0	46.42	3600.00
	25	0.57	1.15	0	51.62	3600.00
	30	0.37	0.82	0	62.04	3600.00
(3)	10	1.64	3.27	100	0.00	1.55
	15	2.40	2.61	100	0.00	42.22
	20	1.77	2.93	75	6.25	1958.21
	25	1.37	2.06	0	52.45	3600.00
	30	1.73	2.21	0	75.22	3600.00

show the experimental results with $m = 2$ and $m = 3$, respectively.

In Tables 3 and 4, the gaps between solutions by the MILP and Algorithm H are 0.61% and 0.77% on average, respectively. It is worth noting that most instances are not solved to optimality by the MILP within the computation time limit when $n \geq 25$. When $m = 2$ or $m = 3$, compared with Algorithm H, the WSPT rule shows good performance in some cases with Scenarios (1) and (2), but for Scenario (3), the gaps are not negligible; larger than 2.4% on average.

In addition, we conducted numerical experiments to evaluate the performance of Algorithm H under extreme conditions. Specifically, we considered parameters $\alpha = 1$, $\beta = 10$, $\gamma = 0.99$, and $\zeta = 10$, which are more extreme than those in Scenario (3) of Table 1. For $n = 20$, the average gaps between MILP and Algorithm H with m of 1, 2, and 3 were 1.3%, 2.08%, and 2.22%, respectively. Although these gaps are slightly larger than those reported in Tables 2–4, the differences are not substantial. Similar results were observed for other values of n . Consequently, based on the numerical results in

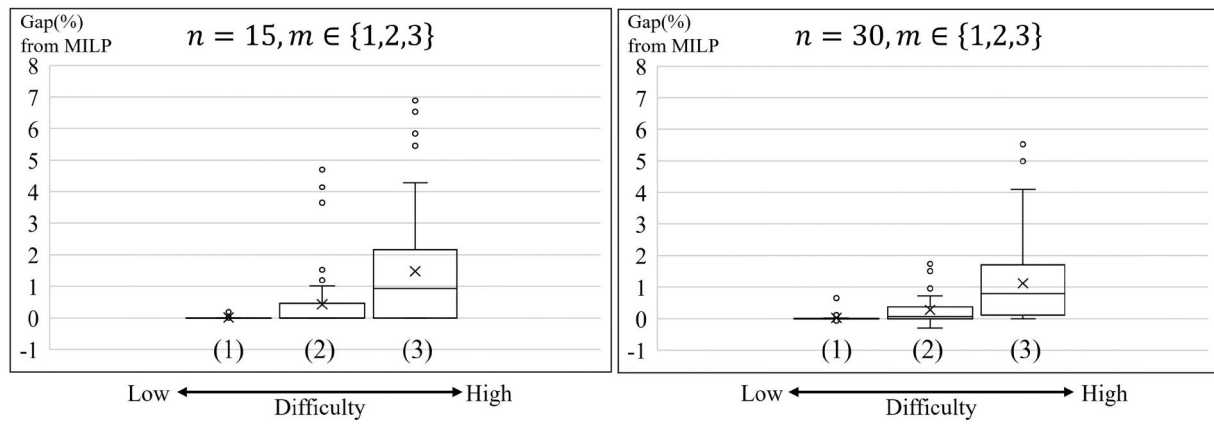


Figure 2. Box plots of the gaps between the MILP and Algorithm H.

Tables 2–4 and the additional tests under extreme cases, it can be confidently stated that Algorithm H provides highly acceptable solutions and consistently outperforms the WSPT rule.

Lastly, Figure 2 validates our experimental design, which assumes three scenarios with varying levels of difficulty. The figure displays box plots that illustrate the gaps between the solutions obtained by MILP and Algorithm H (as defined in (27)), grouped by $n \in \{15, 30\}$ with $m \in \{1, 2, 3\}$. Scenarios (1) through (3) correspond to increasing levels of difficulty, with the y-axis representing the gap(%). As previously discussed, and clearly observed in the figure, the gaps tend to be larger as the difficulty level increases. Nonetheless, even in Scenario (3), Algorithm H demonstrates satisfactory performance.

6. Conclusions and future work

In this paper, we examined a scheduling problem characterized by step-deteriorating job processing times and multiple critical dates in a single machine environment, with the objective of minimizing the TWCT. We analyzed the computational complexities for combinations of various conditions on the critical dates and weights. Additionally, we developed a heuristic based on the WSPT rule. To evaluate our heuristic, we generated numerical instances based on the difficulty level defined by parameters for periods between consecutive critical dates, deterioration rates, basic processing times, and weights. We then investigated how the performance of the proposed heuristic varied across the different levels of difficulty.

For future research, it would be interesting to analyze the exact complexity of cases with a fixed number of critical dates and to identify conditions under which the problem becomes polynomially solvable.

Disclosure statement

No potential conflict of interest was reported by the author(s).

ORCID

Jun-Ho Lee  <http://orcid.org/0000-0001-7693-8896>

References

- Bajestani, M. A., Banjevic, D., & Beck, J. C. (2014). Integrated maintenance planning and production scheduling with Markovian deteriorating machine conditions. *International Journal of Production Research*, 52(24), 7377–7400. <https://doi.org/10.1080/00207543.2014.931609>
- Browne, S., & Yechiali, U. (1990). Scheduling deteriorating jobs on a single processor. *Operations Research*, 38(3), 495–498. <https://doi.org/10.1287/opre.38.3.495>
- Cheng, T. C. E., He, Y., Hoogeveen, H., Ji, M., & Woeginger, G. J. (2006). Scheduling with step-improving processing times. *Operations Research Letters*, 34(1), 37–40. <https://doi.org/10.1016/j.orl.2005.03.002>
- Cheng, T. C. E., Kravchenko, S. A., & Lin, B. M. T. (2020). Scheduling step-deteriorating jobs to minimize the total completion time. *Computers & Industrial Engineering*, 144, 106329. <https://doi.org/10.1016/j.cie.2020.106329>
- Cheng, T., & Ding, Q. (2001). Single machine scheduling with step-deteriorating processing times. *European Journal of Operational Research*, 134(3), 623–630. [https://doi.org/10.1016/S0377-2217\(00\)00284-8](https://doi.org/10.1016/S0377-2217(00)00284-8)
- Cohen, E., & Shapira, D. (2024). Minimising the makespan on parallel identical machines with log-linear position-dependent processing times. *Journal of the Operational Research Society*, 1–9. <https://doi.org/10.1080/01605682.2024.2382150>
- Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: A guide to the theory of NP-completeness*. Freeman. San Francisco.
- Gawiejnowicz, S. (2020a). A review of four decades of time-dependent scheduling: Main results, new topics, and open problems. *Journal of Scheduling*, 23(1), 3–47. <https://doi.org/10.1007/s10951-019-00630-w>
- Gawiejnowicz, S. (2020b). *Models and algorithms for time-dependent scheduling*. Springer.
- Ghaleb, M., Taghipour, S., Sharifi, M., & Zolfagharinia, H. (2020). Integrated production and maintenance

- scheduling for a single degrading machine with deterioration-based failures. *Computers & Industrial Engineering*, 143, 106432. <https://doi.org/10.1016/j.cie.2020.106432>
- Gupta, J. N., & Gupta, S. K. (1988). Single facility scheduling with nonlinear processing times. *Computers & Industrial Engineering*, 14(4), 387–393. [https://doi.org/10.1016/0360-8352\(88\)90041-1](https://doi.org/10.1016/0360-8352(88)90041-1)
- Jeng, A. A. K., & Lin, B. M. T. (2004). Makespan minimization in single-machine scheduling with step-deteriorating of processing times. *Journal of the Operational Research Society*, 55(3), 247–256. <https://doi.org/10.1057/palgrave.jors.2601693>
- Ji, M., He, Y., & Cheng, T. (2007). A simple linear time algorithm for scheduling with step-improving processing times. *Computers & Operations Research*, 34(8), 2396–2402. <https://doi.org/10.1016/j.cor.2005.09.011>
- Kim, E.-S., & Oron, D. (2015). Minimizing total completion time on a single machine with step improving jobs. *Journal of the Operational Research Society*, 66(9), 1481–1490. <https://doi.org/10.1057/jors.2014.91>
- Kim, H.-J., Kim, E.-S., & Lee, J.-H. (2022). Scheduling of step-improving jobs with an identical improving rate. *Journal of the Operational Research Society*, 73(5), 1127–1136. <https://doi.org/10.1080/01605682.2021.1886616>
- Luo, W., Gu, B., & Lin, G. (2018). Communication scheduling in data gathering networks of heterogeneous sensors with data compression: Algorithms and empirical experiments. *European Journal of Operational Research*, 271(2), 462–473. <https://doi.org/10.1016/j.ejor.2018.05.047>
- Lv, D.-Y., & Wang, J.-B. (2024). No-idle flow shop scheduling with deteriorating jobs and common due date under dominating machines. *Asia-Pacific Journal of Operational Research*. <https://doi.org/10.1142/S0217595924500039>
- Lv, Z.-G., Zhang, L.-H., Wang, X.-Y., & Wang, J.-B. (2024). Single machine scheduling proportionally deteriorating jobs with ready times subject to the total weighted completion time minimization. *Mathematics*, 12(4), 610. <https://doi.org/10.3390/math12040610>
- Ma, R., Meng, L., & Zhang, Y. (2024). Online scheduling problem of incompatible batch processing with deterioration effect and delivery time in the steel rolling process. *Journal of the Operational Research Society*, 1–12. <https://doi.org/10.1080/01605682.2024.2366537>
- Mao, R.-R., Lv, D.-Y., Ren, N., & Wang, J.-B. (2024). Supply chain scheduling with deteriorating jobs and delivery times. *Journal of Applied Mathematics and Computing*, 70(3), 2285–2312. <https://doi.org/10.1007/s12190-024-02052-0>
- Mosheiov, G. (1995). Scheduling jobs with step-deterioration; minimizing makespan on a single- and multi-machine. *Computers & Industrial Engineering*, 28(4), 869–879. [https://doi.org/10.1016/0360-8352\(95\)00006-M](https://doi.org/10.1016/0360-8352(95)00006-M)
- Pei, J., Zhou, Y., Yan, P., & Pardalos, P. M. (2023). A concise guide to scheduling with learning and deteriorating effects. *International Journal of Production Research*, 61(6), 2010–2031. <https://doi.org/10.1080/00207543.2022.2049911>
- Pinedo, M. L. (2012). *Scheduling: Theory, algorithms, and systems*. Springer.
- Sloan, T. W., & Shanthikumar, J. G. (2000). Combined production and maintenance scheduling for a multiple-product, single-machine production system. *Production and Operations Management*, 9(4), 379–399. <https://doi.org/10.1111/j.1937-5956.2000.tb00465.x>
- Strusevich, V. A., & Rustogi, K. (2017). *Scheduling with time-changing effects and rate-modifying activities*. Springer.
- Sundararaghavan, P. S., & Kunnathur, A. S. (1994). Single machine scheduling with start time dependent processing times: Some solvable cases. *European Journal of Operational Research*, 78(3), 394–403. [https://doi.org/10.1016/0377-2217\(94\)90048-5](https://doi.org/10.1016/0377-2217(94)90048-5)
- Wang, J.-B., Wang, Y.-C., Wan, C., Lv, D.-Y., & Zhang, L. (2024). Controllable processing time scheduling with total weighted completion time objective and deteriorating jobs. *Asia-Pacific Journal of Operational Research*, 41(3), 2350026. <https://doi.org/10.1142/S0217595923500264>
- Zhang, L.-H., Geng, X.-N., Xue, J., & Wang, J.-B. (2024). Single machine slack due window assignment and deteriorating jobs. *Journal of Industrial and Management Optimization*, 20(4), 1593–1614. <https://doi.org/10.3934/jimo.2023136>