Scheduling of step-improving jobs with an identical improving rate[†]

Hyun-Jung Kim[1], Eun-Seok Kim[2] and Jun-Ho Lee[3‡]

[1] Department of Industrial & Systems Engineering,
Korea Advanced Institute of Science and Technology, Daejeon 34141, Republic of Korea.

[2] School of Business and Management,
Queen Mary University of London, E1 4NS, United Kingdom.

[3] School of Business,
Chungnam National University, Daejeon 34134, Republic of Korea.

February 4, 2021

[†]This paper has been aceepted for publication in Journal of the Operational Research Society (https://doi.org/10.1080/01605682.2021.1886616).

[‡]Corresponding to: Jun-Ho Lee. E-mail: junholee@cnu.ac.kr

Abstract

Job processing times change over time in real-life production and manufacturing systems due to various factors including machine or worker learning, machine deterioration, production system upgrades or technological shocks. For step-improving processing times, job processing times are reduced by a certain rate if they start to process at, or after, a common critical date, which has wide applicability in real-world settings, such as data gathering networks and production systems with part-time workers. This paper considers single machine scheduling of minimizing total weighted completion time with step-improving jobs. The problem is shown to be intractable. Both exact and heuristic algorithms are developed, and the approximability of the heuristic algorithm is shown for a special case of the problem. Finally, computational experiments show that the proposed algorithms provide very effective and efficient solutions.

Keywords: Scheduling, Step-improving processing times, Dynamic programming, Approximation.

1 Introduction

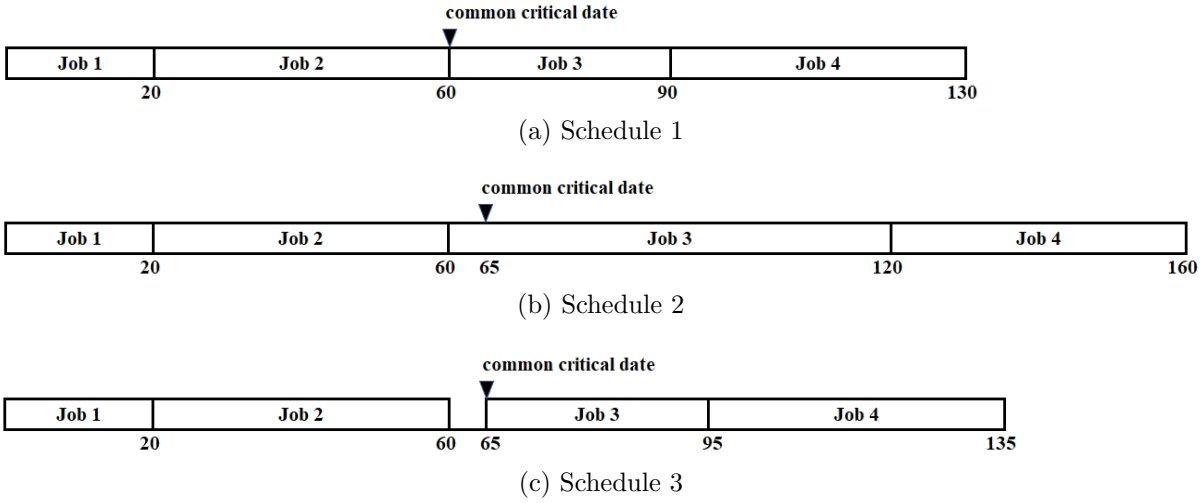
We address a single machine scheduling problem with step-improving processing times in order to minimize the total weighted completion time. In the problem, job processing times are reduced by a certain rate if they start to process at, or after, a common critical date. Such a scheduling problem can be found in data gathering networks where the nodes of the network collect some data and pass them to a single base station (Berlińska 2015, Luo et al. 2018). Each node can compress its data before sending it in order to shorten the transmission time, but the compression process requires a certain amount of time. The base station can communicate with at most one node at a time. Hence, the problem is to determine the nodes that compress their data and the sequence of nodes that use the base station.

Another application can be found in an assembly process of a wiring harness in a company in South Korea. The wiring harness is an assembly of electrical wires, terminals, and connectors, and it is commonly used in automobiles to relay information and electric power throughout a vehicle. The assembly process of the wiring harness consists of cable grapping, clipping, and taping operations, each of which contains several subtasks. Those subtasks are mostly manual and can be performed simultaneously, which implies that the assembly time can be reduced as more workers are assigned to each operation. The company also hires some part-time workers, who work for about one week per month, to satisfy monthly production demand. Then from the time the part-time workers are assigned, the assembly times become shorter depending on the number of assigned workers and their performance levels.

Moreover, this scheduling problem can also be found in many other real-life production and manufacturing systems. When companies are operated with two shifts a day (12 hours per shift), a set of workers assigned to each shift may have different performance levels, which affects the processing times of jobs as well. In addition, job processing times can change

over time due to machine or worker learning, production system upgrades or technological shocks (Cheng et al. 2004, Kim & Oron 2015).

As an illustrative example, we consider four jobs with processing times of 20, 40, 60, and 80, and with weights of 1 for all jobs. Figure 1(a) shows a schedule with the sequence of jobs 1, 2, 3, and 4 when the common critical date is 60 and the reduction rate is 0.5. The processing times of jobs 3 and 4 are 30 and 40, respectively, because they start processing at or after 60. The total weighted completion time of the schedule is 300 ($= 1\times20+1\times60+1\times90+1\times130$). If the common critical date is changed to 65 and the start time of job 3 remains at time 60 as in Figure 1(b), then the makespan becomes 160, and the total weighted completion time is 360. However, if job 3 starts processing at time 65 as in Figure 1(c), then the total weighted completion time is reduced to 310. Hence, inserting an idle time of 5 between 60 and 65 results in a better solution in this example. Therefore, we need to determine not only the sequence of jobs but also the start timing of jobs around the common critical date to reduce the objective measure.



(a) Schedule 1

(b) Schedule 2

(c) Schedule 3

Figure 1: Schedules for an illustrative example

We begin by reviewing the literature in Section 2 and introducing some notation in Section 3. We analyze the complexity of the problem and show that it is NP-hard in Section

4. We then provide a dynamic programming approach and a heuristic algorithm in Section 5. We perform computational experiments to evaluate the performance of the dynamic programming approach and heuristic algorithm in Section 6. Some concluding remarks and directions for future research are given in Section 7.

2 Literature Review

Previous studies that consider time-dependent processing times can be classified into three categories: first, job processing times increase or decrease linearly; second, they follow a piece-wise linear function; third, there are two possible values for the processing times depending on their start times.

In the first category, the processing time of a job changes linearly depending on its star time. Gupta & Gupta (1988) examined a single machine scheduling problem where processing times of jobs follow a monotonically increasing function of their start times with the makespan measure for the first time. They developed an exact optimization algorithm and also heuristic algorithms due to the excessive computation burden of the exact method. Browne & Yechiali (1990) considered a deteriorating job where the job processing time grows at a job-specific rate while waiting for processing. They analyzed different deterioration schemes and derived optimal scheduling policies that minimize the expected makespan. Wu & Lee (2003) addressed a single machine scheduling problem with linear deteriorating jobs in which the processing time of the job follows a nondecreasing function of its starting time. They showed that the linear deteriorating model can be solved with the 0-1 integer programming technique even with the introduction of the machine availability. Wang, Ng & Cheng (2008) considered general linear and proportional linear functions of the job processing time on a single machine. They showed that there exist polynomial time algorithms for the general linear function with the makespan measure and for the proportional linear function

with the total weighted completion time. Li et al. (2011) addressed an optimal due date assignment and a job sequencing simultaneously on a single machine to minimize costs for the earliness, due date assignment and weighted number of tary jobs. In the problem, the processing time of a job is linearly increasing depending on its starting time. Recently, Sun & Geng (2019) examined a single machine scheduling problem with deteriorating effects and machine maintenance to minimize the makespan. In the study, processing times were modeled with a linearly increasing function of the starting time.

In the second category, job processing times follow piece-wise linear functions where the job processing times increase or decrease at a certain point while they remain constant between two consecutive points. Kunnathur & Gupta (1990) presented a model with piece-wise increasing processing times to minimize the makespan, and proposed two optimization algorithms and five heuristic rules. Cheng et al. (2003) considered a single machine scheduling problem where job processing times follow a piece-wise linear nonincreasing function of its start time, and proved that the problem is NP-hard for the objectives of the makespan and total completion time minimization. They also proposed a pseudo-polynomial time algorithm for the makespan and several heuristics for both the total completion time and makespan minimization. Moslehi & Jafari (2010) assumed piece-wise linear deterioration in a single machine with the objective of minimizing the number of tardy jobs, and developed a branch and bound algorithm and a heuristic algorithm.

In the third category, some studies have considered step-functions of job processing times on single machine scheduling. Sundararaghavan & Kunnathur (1994) addressed a single machine scheduling problem where the processing time is a binary function of a common start time due date with the objective of minimizing the total weighted completion time. In the problem, all jobs have the same processing time but their additional processing times required when they start processing after the common critical date are all different. They presented a switching algorithm and a 0-1 quadratic programming formulation. Mosheiov

(1995) considered step-deteriorating jobs with the makespan measure in a single machine, and provided integer programming models and heuristic methods. Cheng & Ding (2001) also assumed step-deteriorating processing times on a single machine with a common critical date. They proved that the problem with the total completion time measure is NP-hard and presented a pseudo-polynomial algorithm for the makespan minimization. Cheng et al. (2006) further considered step-improving job processing times around a common critical date with the makespan measure. They proved that the problem is NP-hard, and provided a pseudo-polynomial time algorithm and an on-line algorithm with the best competitive ratio. Ji et al. (2007) also addressed the same problem in Cheng et al. (2006) by developing a simple linear time off-line approximation algorithm. Biskup & Jahnke (2001) considered a common due date assignment problem with jointly reducible processing times in which all processing times can be reduced by the same proportional amount. Recently, Kim & Oron (2015) addressed the problem in Cheng et al. (2006) with the total completion time measure and proved that the problem is NP-hard. They also provided polynomially solvable cases and a heuristic algorithm for general cases. A similar problem has been solved in the context of data gathering networks (Berlińska 2015, Luo et al. 2018).

Moreover, there are also other papers that consider a different type of the time-dependent processing times, with other variations of classical scheduling models, where job processing times depend on both their start times and positions in a sequence of processing. Yin et al. (2015) considered a new deterioration model where the job processing times depend on both the starting time of jobs and their scheduled position. They showed that the problem with both the makespan and total completion time can be solved in polynomial time. Huang et al. (2010) examined single machine scheduling with the time-dependent deterioration and exponential learning effect, and showed that the makespan minimization problem can be solved with the Shortest Processing Time (SPT) rule. Wang, Ng, Cheng & Liu (2008) considered a time-dependent learning effect for single machine scheduling where the learning effect of

a job is a function of the total normal processing time of the jobs scheduled in front of the job. Rustogi & Strusevich (2014) addressed a combination of time- and position-dependent effects on job processing times subject to rate-modifying activities by considering two objectives, the makespan and total completion time. Recently, Wei (2019) addressed single machine scheduling for general performance measures with learning effects based on the sum-of-processing-time. Wang et al. (2020) examined a single machine scheduling problem with a position-weighted learning effect and job release dates to minimize the total completion time. In the study, logarithm functions were used to model the learning effects. Reviews on scheduling with time-dependent processing times can be found in Gawiejnowicz (2020), Wang, Ng & Cheng (2008), Cheng et al. (2004), Alidaee & Womer (1999).

In this paper, we consider a single machine scheduling problem of minimizing the total weighted completion time where job processing times are reduced by a certain rate at, or after, a common critical date. To the best of knowledge, this is the first paper to consider step-improving jobs and a common critical date with the total weighted completion time measure.

3 Problem Description

There are a set of n non-preemptive jobs $N = \{1, \ldots, n\}$ available for processing at time zero. All jobs in N share a common critical date D which affects their processing times. The processing time of job j is specified by a_j and δ with $0 < \delta < 1$. If job j begins processing at some time $t < D$, then its processing time equals a_j; if it starts at some time $t \geq D$, then its processing time is δa_j. The weight of job j is denoted by w_j. The objective of finding a non-preemptive schedule is to minimize the total weighted completion time.

The standard classification scheme for scheduling problems (Graham *et al.* Graham et al. (1979)) is $\alpha_1|\alpha_2|\alpha_3$, where α_1 describes the machine structure, α_2 gives the job characteristics

or restrictive requirements, and α_3 defines the objective function to be minimized. We extend this scheme to provide for the step-improving processing time with an identical improving rate and a common critical date by using $p_j = a_j$ or $p_j = \delta a_j$ and $d_j = D$ in the α_2 field. Our problem can be denoted as $1|p_j = a_j$ or $p_j = \delta a_j, d_j = D|\sum w_j C_j$.

A schedule σ is an assignment of the jobs in N to the single machine such that each job receives an appropriate amount of processing time, and no two jobs can be processed on the single machine at the same time. Let $S_j(\sigma)$ and $C_j(\sigma)$ denote the start and finish times of job j in schedule σ, respectively. We represent $S_j(\sigma)$ as S_j and $C_j(\sigma)$ as C_j when schedule σ is clear from the context. Let σ^* represent the optimal schedule, i.e., the one which minimizes the total weighted completion time, and let $\sum w_j C_j(\sigma^*)$ indicate the minimum cost associated with this schedule.

4 Complexity Results

This section studies the complexity issue of the considered problem.

Theorem 1 *The problem* $1|p_j = a_j$ *or* $p_j = \delta a_j, d_j = D|\sum w_j C_j$ *is NP-hard.*

Proof. A reduction method is used from the following problem, which is known to be *NP*-complete.

Partition (Garey & Johnson (1979)): Given positive integers $x_1, x_2, \ldots, x_t$, does there exist a set $X \subseteq T = \{1, 2, \ldots, t\}$ such that $\sum_{j \in X} x_j = \sum_{j \in T \setminus X} x_j$?

Assume without loss of generality that $r > 3t$ where $r = \sum_{j \in T} x_j/2$. If not, then we can multiply each partition element and partition size by $3t$ without changing the solution of the problem.

Consider the following instance of $1|p_j = a_j$ or $p_j = \delta a_j, d_j = D|\sum w_j C_j$, called instance I:

$$
\begin{aligned}
n &= t+1, \\
a_j &= 2x_j, \quad j = 1, \ldots, t, \\
a_{t+1} &= 2r^3, \\
w_j &= 1, \quad j = 1, \ldots, t, \\
w_{t+1} &= r^2, \\
\delta &= 0.5, \\
D &= 2r.
\end{aligned}
$$

In the following, we prove that there exists a schedule for this instance of $1|p_j = a_j$ or $p_j = \delta a_j, d_j = D|\sum w_j C_j$ with $\sum w_j C_j \leq K$ if and only if there exists a solution to a partition problem where $K = r^5 + 3r^3 + 3tr$.

($\Leftarrow$) Let X denote a solution to the partition problem. Consider a schedule σ constructed as follows: The jobs in X are scheduled without idle time from time zero; the jobs in $N \setminus X$ are scheduled without idle time from time D; job $t+1$ is scheduled at time $3r$. Thus,

$$
\begin{aligned}
\sum_{j=1}^{t+1} w_j C_j &= \sum_{j=1}^{t} w_j C_j + w_{t+1} C_{t+1} \\
&\leq 3tr + r^2 \left(3r + r^3\right) \\
&= r^5 + 3r^3 + 3tr \\
&= K.
\end{aligned}
$$

The first inequality follows because $w_j = 1$ and $C_j \leq 3r$ for $j = 1, \ldots, t$ and $C_{t+1} = S_{t+1} + \delta a_{t+1} = 3r + r^3$. This implies that there exists a feasible schedule with $\sum w_j C_j \leq K$ for instance I.

($\Rightarrow$) We first show that job $t+1$ is the last job to be processed in an optimal solution σ^*. If job $t+1$ starts before D, then $w_{t+1} C_{t+1} \geq 2r^5 > K$. Thus, job $t+1$ starts at, or after, D. For the jobs starting at, or after, D, the WSPT (Weighted Shortest Processing Time) rule is optimal (Smith (1956)), which implies that job $t+1$ must be the last job in σ^* because $a_{t+1}/w_{t+1} \geq a_j/w_j$ for $j = 1, \ldots, t$. Let X denote a set of jobs that starts before D in the

optimal solution σ^*. If $\sum_{j\in X} a_j < 2r$, then

$$\begin{aligned}\sum_{j=1}^{t+1} w_j C_j &\geq w_{t+1}C_{t+1} \\ &= w_{t+1}\left(D + \delta\left(4r - \sum_{j\in X} a_j\right) + \delta a_{t+1}\right) \\ &\geq r^5 + 3r^3 + r^2 \\ &> K.\end{aligned}$$

The second and the last inequalities follow because $\sum_{j\in X} a_j \leq 2r-2$ and $r > 3t$, respectively. If $\sum_{j\in X} a_j > 2r$, then

$$\begin{aligned}\sum_{j=1}^{t+1} w_j C_j &\geq w_{t+1}C_{t+1} \\ &= w_{t+1}\left(\sum_{j\in X} a_j + \delta\left(4r - \sum_{j\in X} a_j\right) + \delta a_{t+1}\right) \\ &\geq r^5 + 3r^3 + r^2 \\ &> K.\end{aligned}$$

The second and the last inequalities follow because $\sum_{j\in X} a_j \geq 2r+2$ and $r > 3t$, respectively. Therefore, $\sum_{j\in X} a_j = 2r$ if $\sum w_j C_j(\sigma^*) \leq K$, which implies that X is a solution to the partition problem. □

As a result of the proof of NP-hardness, it can be said that no polynomial time exact algorithm may exist for the problem $1|p_j = a_j$ or $p_j = \delta a_j, d_j = D|\sum w_j C_j$ unless $P = NP$.

5 Scheduling algorithms

In this section, we propose scheduling algorithms for the problem $1|p_j = a_j$ or $p_j = \delta a_j, d_j = D|\sum w_j C_j$.

5.1 The exact algorithm

Without loss of generality, the jobs are assumed to be indexed in non-decreasing order of a_j/w_j. We define $f(j,t,u)$ to be the minimum total weighted completion time of jobs $1,\ldots,j$,

where t denotes the total processing time of jobs starting before D, and u denotes the total weight of jobs starting before D. To calculate $f(j,t,u)$, there are two cases to consider: either job j starts before D, or job j starts at, or after, D. Using $f_i(j,t,u)$ for $i=1,2$ to denote such two cases, respectively, then we have

$$f(j,t,u) = \min_{i=1,2} f_i(j,t,u).$$

Each $f_i(j,t,u)$ can be obtained by a dynamic programming recursion as follows.

Case 1: job j starts its processing before D.

For the jobs starting before D, the WSPT rule is optimal (Smith Smith (1956)), which implies that job j is the last of all jobs starting before D. If the completion time of job j is less than D, then the completion times of jobs $1,\ldots,j-1$ remain the same. If the completion time of job j is greater than D, then scheduling job j increases the completion times of the jobs starting at, or after, D by $C_j - D$. Moreover, note that the completion time of job j must be less than $D+\delta a_j$. Otherwise, it is optimal that job j starts at time D. Hence,

$$f_1(j,t,u) = \begin{cases} f(j-1,t-a_j,u-w_j) + w_j t & \text{if } t \le D, \\ f(j-1,t-a_j,u-w_j) + w_j t + \left(\sum_{i=1}^{j} w_i - u\right)(t-D) & \text{if } D < t < D+\delta a_j, \\ +\infty & \text{if } t \ge D+\delta a_j. \end{cases}$$

Case 2: job j starts its processing at, or after, D.

For the jobs starting at, or after, D, the WSPT rule is optimal, which implies that job j is the last of jobs $1,\ldots,j$. Hence,

$$f_2(j,t,u) = f(j-1,t,u) + w_j\left(\max\{t,D\} + \delta\left(b_j - t\right)\right).$$

The initial conditions for $f(0,t,u)$ are $f(0,0,0)=0$ and $f(0,t,u)=+\infty$ for $t>0$ and $u>0$.

Theorem 2 *The minimum total completion time is given by*

$$\min\{f(n,t,u) \mid t = 0,\ldots,\sum_{j=1}^{n} a_j \text{ and } u = 0,\ldots,\sum_{j=1}^{n} w_j\},$$

and the optimal schedule can be found by backtracking.

The time complexity of the dynamic programming is in $O(n\sum_{j=1}^{n} a_j \sum_{j=1}^{n} w_j)$ because j is bounded by n, and t and u are bounded by $\sum_{j=1}^{n} a_j$ and $\sum_{j=1}^{n} w_j$, respectively. This is a pseudo-polynomial algorithm, and since the problem is NP-hard, it is unlikely to solve the problem in polynomial time unless $P = NP$.

5.2 A heuristic algorithm

This section proposes a simple heuristic based on the WSPT rule for the problem. We present a formal description of the heuristic algorithm as follows.

Algorithm A

1. Re-order the jobs such that $a_1/w_1 \leq a_2/w_2 \leq \cdots \leq a_n/w_n$. Determine $k = \max\{j|\sum_{i<j} a_i \leq D\}$.
2. If $\sum_{j=1}^{k} a_j \leq D + \delta a_k$, then schedule the jobs in non-decreasing order of a_j/w_j from time zero without idle time. Otherwise, schedule the jobs $1,\ldots,k-1$ in this order from time zero and the jobs $k,\ldots,n$ in this order from time D.

In order to examine the performance of Algorithm A, we define a preemptive schedule where a job that starts before D and finishes after D is interrupted and resumed at D, and all other jobs are processed without interruption. Let σ_{pmt}^{A} denote a preemptive schedule of a schedule found by Algorithm A: the jobs are scheduled in non-decreasing order of a_j/w_j from time zero without idle time; job k is interrupted and resumed at D; all other jobs $j \in N \setminus \{k\}$ are processed without interruption.

Lemma 1 *If the jobs have agreeable weights, that is, $a_k < a_l$ implies $w_k \geq w_l$, then σ^A_{pmt} is an optimal preemptive schedule.*

Proof. For ease of analysis, the jobs are assumed to be indexed in non-decreasing order of a_j/w_j without loss of generality. For any given optimal preemptive schedule, let k denote the last job starting processing before D, and there are two sets of jobs: one is a set of jobs scheduled before job k, denoted as $\mathcal{J}_1$; another is a set of jobs scheduled after D, denoted as $\mathcal{J}_2$. Note that the jobs in each set of jobs, $\mathcal{J}_1$ and $\mathcal{J}_2$ must be in the WSPT order in an optimal preemptive schedule. Let i and l denote the last job in $\mathcal{J}_1$ and the first job in $\mathcal{J}_2$, respectively.

We show that $a_i/w_i \leq a_k/w_k \leq a_l/w_l$ in an optimal preemptive schedule. Firstly, suppose that there exists an optimal preemptive schedule σ_{pmt} where $a_i/w_i > a_k/w_k$. Construct a preemptive schedule σ'_{pmt} by exchanging jobs i and k where $\tau = S_i(\sigma_{pmt})$, as shown in Figure 2.

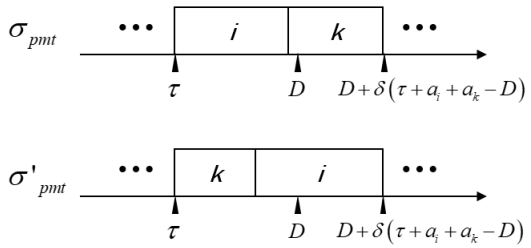


Figure 2: Structure of schedules σ_{pmt} and σ'_{pmt}

Then,

$$\begin{aligned} C_i(\sigma_{pmt}) &= \tau + a_i, \\ C_k(\sigma_{pmt}) &= D + \delta(\tau + a_i + a_k - D), \\ C_i(\sigma'_{pmt}) &= D + \delta(\tau + a_i + a_k - D), \\ C_k(\sigma'_{pmt}) &= \tau + a_k. \end{aligned}$$

Since $C_j(\sigma_{pmt}) = C_j\left(\sigma'_{pmt}\right)$ for all $j \in N \setminus \{i,k\}$,

$$\begin{aligned}
\sum_{j=1}^{n} w_j C_j\left(\sigma'_{pmt}\right) - \sum_{j=1}^{n} w_j C_j(\sigma_{pmt}) &= w_i(D + \delta(\tau + a_i + a_k - D) - \tau - a_i) \\
&\quad - w_k(D + \delta(\tau + a_i + a_k - D) - \tau - a_k) \\
&\leq (w_i - w_k)(D + \delta(\tau + a_i + a_k - D) - \tau - a_i) \\
&\leq 0.
\end{aligned}$$

The first and second inequality holds that $a_i \geq a_k$ and $w_i \leq w_k$, respectively, because the jobs i and k have agreeable weights. This contradicts that σ_{pmt} is optimal.

To complete the proof, suppose that there exists an optimal preemptive schedule σ_{pmt} where $a_k/w_k > a_l/w_l$. Construct a preemptive schedule σ'_{pmt} by exchanging jobs k and l where $\tau = S_k(\sigma_{pmt})$, as shown in Figure 3.

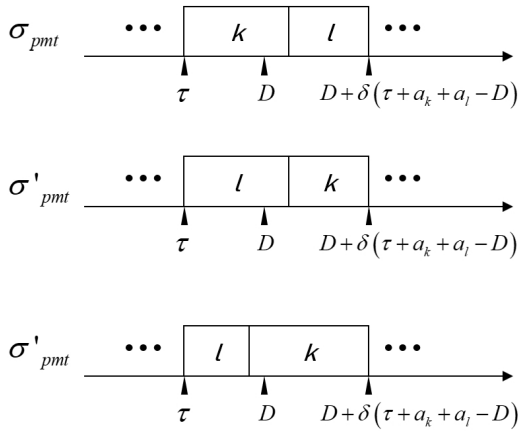


Figure 3: Structure of schedules σ_{pmt} and σ'_{pmt}

Note that a job that starts before D and finishes after D is interrupted and resumed at

D in a preemptive schedule. Thus,

$$\begin{aligned}
C_k(\sigma_{pmt}) &= D + \delta(\tau + a_k - D), \\
C_l(\sigma_{pmt}) &= D + \delta(\tau + a_k + a_l - D), \\
C_k(\sigma'_{pmt}) &= D + \delta(\tau + a_k + a_l - D), \\
C_l(\sigma'_{pmt}) &= \begin{cases} D + \delta(\tau + a_l - D) & \text{if } \tau + a_l > D, \\ \tau + a_l & \text{if } \tau + a_l \le D. \end{cases}
\end{aligned}$$

Note that $C_j(\sigma_{pmt}) = C_j(\sigma'_{pmt})$ for all $j \in N \setminus \{k, l\}$. If $\tau + a_l > D$, then

$$\sum_{j=1}^{n} w_j C_j(\sigma'_{pmt}) - \sum_{j=1}^{n} w_j C_j(\sigma_{pmt}) = \delta(w_k a_l - w_l a_k) < 0.$$

If $\tau + a_l \le D$, then

$$\begin{aligned}
\sum_{j=1}^{n} w_j C_j(\sigma'_{pmt}) - \sum_{j=1}^{n} w_j C_j(\sigma_{pmt}) &\le \delta w_k a_l + w_l(\tau + a_l - D) \\
&\quad - \delta w_l(\tau + a_k + a_l - D) \\
&= \delta w_k a_l + (1-\delta) w_l(\tau + a_l - D) - \delta w_l a_k \\
&\le \delta(w_k a_l - w_l a_k) < 0.
\end{aligned}$$

The second inequality holds because $\tau + a_l - D \le 0$. This contradicts that σ_{pmt} is optimal. □

We now show the approximability of Algorithm A when the jobs have agreeable weights, that is, $a_k < a_l$ implies $w_k \ge w_l$. An algorithm is called as α-approximation algorithm for a problem if for every instance of the problem it can find a solution within a factor α of the optimum solution.

Theorem 3 *If the jobs have agreeable weights, that is, $a_k < a_l$ implies $w_k \ge w_l$, then Algorithm A is a $(2-\delta)$-approximation algorithm.*

Proof. Let σ^A denote a schedule found by Algorithm A. As in the formal description of Algorithm A, let k denote the last job that starts processing before or at D and $b_j = \sum_{i=1}^{j} a_i$.

From the result of Lemma 1, in an optimal preemptive schedule σ^A_{pmt},

$$\begin{aligned} C_j(\sigma^A_{pmt}) &= b_j \text{ for } j = 1, \ldots, k-1, \\ C_j(\sigma^A_{pmt}) &= D + \delta(b_j - D) = (1-\delta)D + \delta b_j \text{ for } j = k, \ldots, n. \end{aligned}$$

Therefore,

$$\begin{aligned} \sum_{j=1}^{n} w_j C_j(\sigma^A_{pmt}) &= \sum_{j=1}^{k-1} w_j b_j + \sum_{j=k}^{n} w_j \left((1-\delta)D + \delta b_j\right) \\ &= \sum_{j=1}^{k-1} w_j b_j + \delta \sum_{j=k}^{n} w_j b_j + (1-\delta)D \sum_{j=k}^{n} w_j \\ &\geq \delta \sum_{j=k}^{n} w_j b_j \\ &\geq \delta \sum_{j=k}^{n} w_j b_k. \end{aligned}$$

The last inequality follows because $b_j \geq b_k$ for $j = k, \ldots, n$.

We now examine a schedule found by Algorithm A. There are two cases to consider: either job k starts before D or job k starts at D in σ^A.

Case 1: job k starts its processing before D in σ^A.

In this case, $C_k(\sigma^A) = b_k$. Thus,

$$\begin{aligned} \sum_{j=1}^{n} w_j C_j(\sigma^A) - \sum_{j=1}^{n} w_j C_j(\sigma^A_{pmt}) &= \left(\sum_{j=k}^{n} w_j\right)\left(b_k - (D + \delta\left(b_k - D\right))\right) \\ &= \left(\sum_{j=k}^{n} w_j\right)(1-\delta)\left(b_k - D\right) \\ &\leq \left(\sum_{j=k}^{n} w_j\right)(1-\delta)\delta a_k. \end{aligned}$$

The last inequality follows because job k starts its processing before D in σ^A which implies that $b_k \leq D + \delta a_k$.

Case 2: job k starts its processing at D in σ^A.

In this case, $C_k(\sigma^A) = D + \delta a_k$. Thus,

$$\begin{aligned} \sum_{j=1}^{n} w_j C_j(\sigma^A) - \sum_{j=1}^{n} w_j C_j(\sigma^A_{pmt}) &= \left(\sum_{j=k}^{n} w_j\right)\delta\left(a_k - b_k + D\right) \\ &\leq \left(\sum_{j=k}^{n} w_j\right)\delta\left(a_k - \delta a_k\right) \\ &\leq \left(\sum_{j=k}^{n} w_j\right)(1-\delta)\delta a_k. \end{aligned}$$

The first inequality follows because job k starts its processing at D in σ^A which implies that $D + \delta a_k \leq b_k$.

Therefore,

$$\begin{aligned}
\frac{\sum_{j=1}^{n} w_j C_j(\sigma^A) - \sum_{j=1}^{n} w_j C_j(\sigma^A_{pmt})}{\sum_{j=1}^{n} w_j C_j(\sigma^A_{pmt})} &\leq \frac{\left(\sum_{j=k}^{n} w_j\right)(1-\delta)\delta a_k}{\left(\sum_{j=k}^{n} w_j\right)\delta b_k} \\
&\leq \frac{(1-\delta)a_k}{b_k} \\
&\leq 1-\delta.
\end{aligned}$$

The last inequality follows because $a_k \leq b_k$. Since $\sum_{j=1}^{n} w_j C_j(\sigma^A_{pmt}) = \sum_{j=1}^{n} w_j C_j(\sigma^*_{pmt}) \leq \sum_{j=1}^{n} w_j C_j(\sigma^*)$,

$$\frac{\sum_{j=1}^{n} w_j C_j(\sigma^A) - \sum_{j=1}^{n} w_j C_j(\sigma^*)}{\sum_{j=1}^{n} w_j C_j(\sigma^*)} \leq \frac{\sum_{j=1}^{n} w_j C_j(\sigma^A) - \sum_{j=1}^{n} w_j C_j(\sigma^A_{pmt})}{\sum_{j=1}^{n} w_j C_j(\sigma^A_{pmt})} \leq 1-\delta.$$

As a result, Algorithm A is a $(2-\delta)$-approximation algorithm. □

For $\delta = 0$, Algorithm A finds an optimal solution, but the approximation factor is 2. This implies that the $(2-\delta)$-approximation algorithm is not tight.

6 Computational study

In this section, we undertake extensive numerical tests to analyze the performance of the exact algorithm (i.e., dynamic programming) and the heuristic algorithm (i.e., Algorithm A), proposed in Sections 5.1 and 5.2, respectively. The algorithms are coded in JAVA with 20GB heap space, and run on a personal computer with Intel Core i7-9700 processor with a 3-GHz clock and 32GB RAM.

We evaluate the performance of the proposed algorithms by considering the impact of: number of jobs (n), processing times (a_j), weights (w_j), the improving rate (δ), and the common critical date (D). The experiments are designed to evaluate: firstly, the computational performance of the exact algorithm; secondly, the quality of solutions found by

the heuristic algorithm. To test the computational performance of the exact algorithm, we let $n \in \{100, 200, 400\}$; $a_j \sim$ DU[1,30], $a_j \sim$ DU[1,50] and $a_j \sim$ DU[1,100]; and $w_j \sim$ DU[1,5], $w_j \sim$ DU[1,10] and $w_j \sim$ DU[1,20], where DU[l, u] is the discrete uniform distribution over the interval [l, u]. Note that the time complexity of the exact algorithm is in $O\left(n \sum_{j=1}^{n} a_j \sum_{j=1}^{n} w_j\right)$. These results are presented in Table 1. To test the quality of solutions found by the heuristic algorithm, we let $n \in \{100, 200, 400\}$; $a_j \sim$ DU[1,30], $a_j \sim$ DU[1,50] and $a_j \sim$ DU[1,100]; $w_j \sim$ DU[1,5], $w_j \sim$ DU[1,10] and $w_j \sim$ DU[1,20]; $\delta \in \{0.2, 0.5, 0.8\}$; and $D = \alpha \sum_{j=1}^{n} a_j$ where $\alpha \in \{0.1, 0.3, 0.5\}$. These results are presented in Table 2. Our performance indicator is the percentage gap between the total weighted completion times from the exact and heuristic algorithms, which is computed by $100 \times (z^A - z^E)/z^E$ where z^A and z^E are the total weighted completion times of the heuristic algorithm and exact algorithm, respectively. For each condition, the table entry is the average of 20 instances that are randomly generated. Times are given in seconds.

In Table 1, we can observe that the computation time of the exact algorithm rapidly increases as n, a_j and w_j increase because the time complexity of the exact algorithm is in $O\left(n \sum_{j=1}^{n} a_j \sum_{j=1}^{n} w_j\right)$. However, the exact algorithm finds an optimal solution in reasonable times for up to 400 jobs. In particular, the average computation times for $n = 100$, 200 and 400 are 7.6 s, 76.9 s and 1814.8 s, respectively.

In Table 2, we can observe that the heuristic algorithm finds near-optimal solutions. The average gap between solutions from the heuristic and exact algorithms is only 0.12%. The maximum gap is 1.09% which occurs for $n = 100$, $a_j \sim$ DU[1,50], $w_j \sim$ DU[1,20], $\alpha = 0.1$ and $\delta = 0.5$. It is worth noting that the average computation time of the heuristic algorithm is less than 0.001 s. Therefore, when the problem size is very large, it is sensible to use the heuristic algorithm as an effective alternative.

In Table 2, we can also observe that n, δ, and α affect the performance of the heuristic algorithm. Firstly, the average gap tends to decrease as the number of jobs increases as

Table 1: Computational performance of the exact algorithm.

n	a_j	w_j	Average Computation Times (s)	Maximum Computation Times (s)
100	DU[1,30]	DU[1,5]	2.1	3.2
		DU[1,10]	2.2	2.8
		DU[1,20]	7.8	9.1
	DU[1,50]	DU[1,5]	2.0	2.4
		DU[1,10]	3.7	4.6
		DU[1,20]	7.3	10.8
	DU[1,100]	DU[1,5]	7.0	8.4
		DU[1,10]	12.3	13.9
		DU[1,20]	23.7	29.8
200	DU[1,30]	DU[1,5]	18.2	21.1
		DU[1,10]	34.2	46.8
		DU[1,20]	64.3	80.0
	DU[1,50]	DU[1,5]	30.4	37.2
		DU[1,10]	54.9	70.7
		DU[1,20]	106.8	125.2
	DU[1,100]	DU[1,5]	61.1	74.4
		DU[1,10]	111.5	154.8
		DU[1,20]	210.5	246.0
400	DU[1,30]	DU[1,5]	444.5	537.1
		DU[1,10]	798.0	907.9
		DU[1,20]	1510.5	1647.8
	DU[1,50]	DU[1,5]	728.3	857.2
		DU[1,10]	1325.1	1431.2
		DU[1,20]	2492.1	2776.7
	DU[1,100]	DU[1,5]	1434.9	1667.4
		DU[1,10]	2626.4	2948.3
		DU[1,20]	4973.0	5431.6

shown in Figure 4(a). It is because the denominator of the gap, z^E, becomes larger as n increases whereas the numerator of the gap, $z^A - z^E$, coming from the idle time near the common critical date is relatively constant. Secondly, as shown in Figure 4(b), the average gap is maximized when $\delta = 0.5$. When δ is close to 0 or 1, the problem becomes similar to traditional single machine scheduling problems with constant processing times, and hence the heuristic algorithm based on WSPT works well. Lastly, the average gap tends to decrease as the common critical date, D (or equivalently α), increases as shown in Figure 4(c). The

Table 2: Performance of the heuristic algorithm.

n	a_j	w_j	$\delta = 0.2$			$\delta = 0.5$			$\delta = 0.8$		
			$\alpha = 0.1$	$\alpha = 0.3$	$\alpha = 0.5$	$\alpha = 0.1$	$\alpha = 0.3$	$\alpha = 0.5$	$\alpha = 0.1$	$\alpha = 0.3$	$\alpha = 0.5$
			Avg(%)	Avg(%)	Avg(%)	Avg(%)	Avg(%)	Avg(%)	Avg(%)	Avg(%)	Avg(%)
100	DU[1,30]	DU[1,5]	0.29	0.24	0.13	0.32	0.27	0.21	0.16	0.09	0.08
		DU[1,10]	0.36	0.24	0.10	0.37	0.31	0.12	0.16	0.17	0.07
		DU[1,20]	0.34	0.20	0.10	0.27	0.24	0.14	0.15	0.12	0.09
	DU[1,50]	DU[1,5]	0.32	0.21	0.11	0.36	0.22	0.14	0.18	0.09	0.09
		DU[1,10]	0.33	0.22	0.10	0.36	0.26	0.14	0.16	0.13	0.09
		DU[1,20]	0.35	0.18	0.09	0.44	0.29	0.16	0.22	0.17	0.09
	DU[1,100]	DU[1,5]	0.34	0.21	0.08	0.37	0.29	0.11	0.16	0.16	0.07
		DU[1,10]	0.34	0.21	0.09	0.30	0.25	0.15	0.14	0.15	0.08
		DU[1,20]	0.38	0.20	0.11	0.43	0.27	0.17	0.19	0.18	0.08
200	DU[1,30]	DU[1,5]	0.17	0.10	0.06	0.19	0.14	0.08	0.09	0.07	0.04
		DU[1,10]	0.19	0.11	0.06	0.20	0.15	0.09	0.10	0.09	0.04
		DU[1,20]	0.17	0.10	0.07	0.14	0.15	0.09	0.06	0.09	0.04
	DU[1,50]	DU[1,5]	0.16	0.13	0.04	0.14	0.16	0.05	0.07	0.08	0.03
		DU[1,10]	0.22	0.10	0.05	0.17	0.12	0.07	0.06	0.07	0.03
		DU[1,20]	0.16	0.13	0.05	0.17	0.14	0.08	0.09	0.07	0.04
	DU[1,100]	DU[1,5]	0.17	0.11	0.07	0.20	0.16	0.08	0.09	0.09	0.05
		DU[1,10]	0.18	0.10	0.06	0.17	0.15	0.08	0.09	0.08	0.04
		DU[1,20]	0.16	0.12	0.06	0.18	0.12	0.07	0.10	0.06	0.04
400	DU[1,30]	DU[1,5]	0.09	0.07	0.02	0.10	0.08	0.03	0.04	0.03	0.02
		DU[1,10]	0.09	0.05	0.03	0.09	0.07	0.04	0.04	0.04	0.02
		DU[1,20]	0.08	0.05	0.03	0.08	0.06	0.04	0.05	0.04	0.02
	DU[1,50]	DU[1,5]	0.07	0.06	0.03	0.08	0.06	0.05	0.03	0.04	0.03
		DU[1,10]	0.08	0.04	0.03	0.11	0.06	0.04	0.05	0.05	0.02
		DU[1,20]	0.09	0.06	0.03	0.09	0.08	0.04	0.03	0.04	0.03
	DU[1,100]	DU[1,5]	0.10	0.05	0.02	0.09	0.06	0.03	0.05	0.03	0.03
		DU[1,10]	0.11	0.06	0.03	0.12	0.09	0.04	0.05	0.05	0.02
		DU[1,20]	0.09	0.06	0.03	0.12	0.08	0.04	0.06	0.04	0.02
Average			0.20	0.13	0.06	0.21	0.16	0.09	0.10	0.09	0.05

heuristic algorithm inserts an idle time right before D if the condition, $\sum_{j=1}^{k} a_j \leq D + \delta a_k$, is not satisfied. If D is small, then the inserted idle time before D has a significant impact on the total weighted completion time because a large number of jobs assigned at, or after, D are delayed by the idle time.

In addition, we carried out an one-way analysis of variance (ANOVA) to see whether the differences between average gaps with different n, δ, and α are statistically significant. Specifically, we consider the following three cases: (1) differences between the average gaps with n of 100, 200, and 400, (2) differences between the average gaps with δ of 0.2, 0.5,

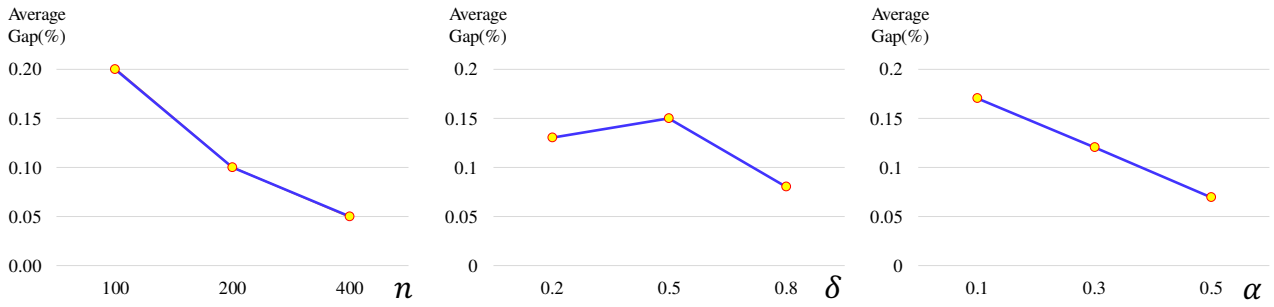


Figure 4: Performance of the heuristic algorithm with different n, δ and α

and 0.8, (3) differences between the average gaps with α of 0.1, 0.3, and 0.5. P-values from ANOVA are very small as shown in Table 3, and we can conclude that the differences between average gaps with different n, δ, and α are statistically significant.

Table 3: ANOVA results

Description	ANOVA P-value
Differences between average gaps with n of 100, 200, and 400	2.99×10^{-34}
Differences between average gaps with δ of 0.2, 0.5, and 0.8	1.61×10^{-7}
Differences between average gaps with α of 0.1, 0.3, and 0.5	7.15×10^{-14}

7 Conclusions

This paper studies the single machine scheduling problem of minimizing total weighted completion time with step-improving processing time jobs. The job processing times are reduced by a certain rate if they start to process at, or after, a common critical date. We establish that the problem is NP-hard. For the problem, we develop the pseudo-polynomial exact algorithm based on dynamic programming and the simple heuristic algorithm based on the WSPT rule. In order to evaluate the effectiveness and efficiency of the proposed algorithms, computational experiments are carried out. The experiment results show that the exact algorithm can find an optimal solution in reasonable times for up to 400 jobs and that the

heuristic algorithm provides near-optimal solutions with the average gap of 0.12% in short computation times.

Future research could extend our results to job-dependent critical dates. Also, it would be interesting to consider step-improving processing times with different scheduling criteria in various multiple machine environments. Finally, the complexity of the unweighted model for the considered problem is still open.

Acknowledgement

This work was supported by Global Research Network program through the Ministry of Education of the Republic of Korea and the National Research Foundation of Korea (NRF-2019S1A2A2031006).

References

Alidaee, B. & Womer, N. K. (1999), 'Scheduling with time dependent processing times: Review and extensions', *Journal of the Operational Research Society* **50**(1), 711 – 720.

Berlińska, J. (2015), 'Scheduling for data gathering networks with data compression', *European Journal of Operational Research* **246**(3), 744–749.

Biskup, D. & Jahnke, H. (2001), 'Common due date assignment for scheduling on a singlemachine with jointly reducible processing times', *International Journal of Production Economics* **69**(3), 317 – 322.

Browne, S. & Yechiali, U. (1990), 'Scheduling deteriorating jobs on a single processor', *Operations Research* **38**(3), 495–498.

Cheng, T. C. E., Ding, Q., Kovalyov, M. Y., Bachman, A. & Janiak, A. (2003), 'Scheduling jobs with piecewise linear decreasing processing times', *Naval Research Logistics* **50**(6), 531–554.

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), 1 – 13.

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.

Cheng, T. & Ding, Q. (2001), 'Single machine scheduling with step-deteriorating processing times', *European Journal of Operational Research* **134**(3), 623 – 630.

Garey, M. R. & Johnson, D. S. (1979), *Computers and Intractability: A guide to the theory of NP-completeness*, W. H. Freeman and Company, New York, NY, USA.

Gawiejnowicz, S. (2020), 'A review of four decades of time-dependent scheduling: main results, new topics, and open problems', *Journal of Scheduling* **23**(1), 3–47.

Graham, R. L., Lawler, E. L., Lenstra, J. K. & Rinnooy-Kan, A. (1979), 'Optimization and approximation in deterministic machine scheduling: A survey', *Annals of Discrete Mathematics* **15**, 287–326.

Gupta, J. N. D. & Gupta, S. K. (1988), 'Single facility scheduling with nonlinear processing times', *Computers & Industrial Engineering* **14**(4), 387–393.

Huang, X., Wang, J.-B., Wang, L.-Y., Gao, W.-J. & Wang, X.-R. (2010), 'Single machine scheduling with time-dependent deterioration and exponential learning effect', *Computers & Industrial Engineering* **58**(1), 58 – 63.

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.

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.

Kunnathur, A. S. & Gupta, S. K. (1990), 'Minimizing the makespan with late start penalties added to processing times in a single facility scheduling problem', *European Journal of Operational Research* **47**(1), 56 – 64.

Li, S., Ng, C. T. & Yuan, J. (2011), 'Scheduling deteriorating jobs with CON/SLK due date assignment on a single machine', *International Journal of Production Economics* **131**(2), 747 – 751.

Luo, W., Gu, B. & Lin, G. (2018), 'Communication scheduling in data gathering neworks of heterogeneous sensors with data compression: Algorithms and empirical experiments', *European Journal of Operational Research* **271**(2), 462–473.

Mosheiov, G. (1995), 'Scheduling jobs with step-deterioration; minimizing makespan on a single- and multi-machine', *Computers & Industrial Engineering* **28**(4), 869 – 879.

Moslehi, G. & Jafari, A. (2010), 'Minimizing the number of tardy jobs under piecewise-linear deterioration', *Computers & Industrial Engineering* **59**(4), 573 – 584.

Rustogi, K. & Strusevich, V. A. (2014), 'Combining time and position dependent effects on a single machine subject to rate-modifying activities', *Omega* **42**(1), 166 – 178.

Smith, W. E. (1956), 'Various optimizer for single stage production', *Naval Research Logistics Quarterly* **3**, 59–66.

Sun, X. & Geng, X.-N. (2019), 'Single-machine scheduling with deteriorating effects and machine maintenance', *International Journal of Production Research* **57**(10), 3186–3199.

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.

Wang, J.-B., Gao, M., Wang, J.-J., Liu, L. & He, H. (2020), 'Scheduling with a position-weighted learning effect and job release dates', *Engineering Optimization (DOI: 10.1080/0305215X.2019.1664498)* .

Wang, J.-B., Ng, C. T. & Cheng, T. C. E. (2008), 'Single-machine scheduling with deteriorating jobs under a series–parallel graph constraint', *Computers & Operations Research* **35**(8), 2684 – 2693.

Wang, J.-B., Ng, C. T., Cheng, T. C. E. & Liu, L. L. (2008), 'Single-machine scheduling with a time-dependent learning effect', *International Journal of Production Economics* **111**(2), 802 – 811.

Wei, W. (2019), 'Single machine scheduling with stochastically dependent times', *Journal of Scheduling* **22**, 677–689.

Wu, C.-C. & Lee, W.-C. (2003), 'Scheduling linear deteriorating jobs to minimize makespan with an availability constraint on a single machine', *Information Processing Letters* **87**(2), 89 – 93.

Yin, Y., Wu, W.-H., Cheng, T. & Wu, C.-C. (2015), 'Single-machine scheduling with time-dependent and position-dependent deteriorating jobs', *International Journal of Computer Integrated Manufacturing* **28**(7), 781–790.