2016

Single Machine Scheduling with Job-Dependent Machine Deterioration

Wenchang Luo
University of Alberta, wenchang@ualberta.ca

Xu Yao
University of Alberta, xu2@ualberta.ca

Weitian Tong
Georgia Southern University, wtong@georgiasouthern.edu

Guohui Lin
University of Alberta, guohui@ualberta.ca

Single Machine Scheduling with Job-Dependent Machine Deterioration*

Wenchang Luo[1], Yao Xu[2], Weitian Tong[3], and Guohui Lin[4]

1 **Faculty of Science, Ningbo University. Ningbo, Zhejiang 315211, China; and Department of Computing Science, University of Alberta. Edmonton, Alberta T6G 2E8, Canada**
wenchang@ualberta.ca

2 **Department of Computing Science, University of Alberta. Edmonton, Alberta T6G 2E8, Canada**
xu2,guohui@ualberta.ca

3 **Department of Computer Sciences, Georgia Southern University. Statesboro, Georgia 30460, USA**
wtong@georgiasouthern.edu

4 **Department of Computing Science, University of Alberta. Edmonton, Alberta T6G 2E8, Canada**
guohui@ualberta.ca

Abstract

We consider the single machine scheduling problem with job-dependent machine deterioration. In the problem, we are given a single machine with an initial non-negative maintenance level, and a set of jobs each with a non-preemptive processing time and a machine deterioration. Such a machine deterioration quantifies the decrement in the machine maintenance level after processing the job. To avoid machine breakdown, one should guarantee a non-negative maintenance level at any time point; and whenever necessary, a maintenance activity must be allocated for restoring the machine maintenance level. The goal of the problem is to schedule the jobs and the maintenance activities such that the total completion time of jobs is minimized. There are two variants of maintenance activities: in the partial maintenance case each activity can be allocated to increase the machine maintenance level to any level not exceeding the maximum; in the full maintenance case every activity must be allocated to increase the machine maintenance level to the maximum. In a recent work, the problem in the full maintenance case has been proven NP-hard; several special cases of the problem in the partial maintenance case were shown solvable in polynomial time, but the complexity of the general problem is left open. In this paper we first prove that the problem in the partial maintenance case is NP-hard, thus settling the open problem; we then design a 2-approximation algorithm.

1998 ACM Subject Classification F.2.2 Sequencing and scheduling, F.1.3 Reducibility and completeness, I.1.2 Analysis of algorithms

Keywords and phrases Scheduling, machine deterioration, maintenance, NP-hard, approximation algorithm

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2016.55

* This work was partially supported by the K. C. Wong Magna Foundation of Ningbo University, the China Scholarship Council (Grant No. 201408330402), the Ningbo Natural Science Foundation (2016A610078) to W. L., the NSERC Canada to G. L., and the Office of the Vice President for Research & Economic Development and Georgia Southern University to W. T.

1 Introduction

In many scheduling problems, processing a job on a machine causes the machine to deteriorate to some extent, and consequently maintenance activities need to be executed in order to restore the machine capacity. Scheduling problems with maintenance activities have been extensively investigated since the work of Lee and Liman [7].

A maintenance activity is normally described by two parameters, the starting time and the duration. If these two parameters are given beforehand, a maintenance activity is referred to as *fixed*; otherwise it is called *flexible*. Various scheduling models with fixed maintenance activities, on different machine environments and job characteristics, have been comprehensively surveyed by Schmidt [14], Lee [5], and Ma *et al.* [10].

A number of researchers initiated the work with flexible maintenance activities. Qi *et al.* [13] considered a single machine scheduling problem to simultaneously schedule jobs and maintenance activities, with the objective to minimize the total completion time of jobs. They showed that the problem is NP-hard in the strong sense and proposed heuristics and a branch-and-bound exact algorithm. (Qi [12] later analyzed the worst-case performance ratio for one of the heuristics, *the shortest processing time first* or SPT.) Lee and Chen [6] studied the multiple parallel machines scheduling problem where each machine must be maintained exactly once, with the objective to minimize the total weighted completion time of jobs. They proved the NP-hardness for some special cases and proposed a branch-and-bound exact algorithm based on column generation; the NP-hardness for the general problem is implied. Kubzin and Strusevich [4] considered a two-machine open shop and a two-machine flow shop scheduling problems in which each machine has to be maintained exactly once and the duration of each maintenance depends on its starting time. The objective is to minimize the maximum completion time of all jobs and all maintenance activities. Among others, the authors showed that the open shop problem is polynomial time solvable for quite general functions defining the duration of maintenance in its starting time; they also proved that the flow shop problem is binary NP-hard and presented a fully polynomial time approximation scheme (FPTAS) [4].

Returning to a single machine scheduling problem, Chen [2] studied the periodic maintenance activities of a constant duration not exceeding the available period, with the objective to minimize the maximum completion time of jobs (that is, the *makespan*). The author presented two mixed integer programs and heuristics and conducted computational experiments to examine their performance. Mosheiov and Sarig [11] considered the problem where the machine needs to be maintained prior to a given deadline, with the objective to minimize the total weighted completion time of jobs. They showed the binary NP-hardness and presented a pseudo-polynomial time dynamic programming algorithm and an efficient heuristic. Luo *et al.* [8] investigated a similar variant (to [11]) in which the jobs are weighted and the duration of the maintenance is a nondecreasing function of the starting time (which must be prior to a given deadline). Their objective is to minimize the total weighted completion time of jobs; the authors showed the weak NP-hardness, and for the special case of concave duration function they proposed a $(1 + \sqrt{2}/2 + \epsilon)$-approximation algorithm. Yang and Yang [17] considered a position-dependent aging effect described by a power function under maintenance activities and variable maintenance duration considerations simultaneously; they examined two models with the objective to minimize the makespan, and for each of them they presented a polynomial time algorithm.

Scheduling on two identical parallel machines with periodic maintenance activities was examined by Sun and Li [15], where the authors presented approximation algorithms with

constant performance ratios for minimizing the makespan or minimizing the total completion time of jobs. Xu *et al.* [16] considered the case where the length of time between two consecutive maintenances is bounded; they presented an approximation algorithm for the multiple parallel machines scheduling problem to minimize the completion time of the last maintenance, and for the single machine scheduling problem to minimize the makespan, respectively.

1.1 Problem definition

Considering the machine deterioration in the real world, in a recent work by Bock *et al.* [1], a new scheduling model subject to *job-dependent machine deterioration* is introduced. In this model, the single machine must have a non-negative *maintenance level* (ML) at any time point, specifying its current maintenance state. (A negative maintenance level indicates the machine breakdown, which is prohibited.) We are given a set of jobs $\mathcal{J} = \{J_i, i = 1, 2, \ldots, n\}$, where each job $J_i = (p_i, \delta_i)$ is specified by its non-preemptive *processing time* p_i and *machine deterioration* δ_i. The machine deterioration δ_i quantifies the decrement in the machine maintenance level after processing the job J_i. (That is, if before processing the job J_i the maintenance level is ML, then afterwards the maintenance level reduces to $\mathrm{ML} - \delta_i$ — suggesting that ML has to be at least δ_i in order for the machine to process the job J_i.)

Clearly, to process all the jobs, *maintenance activities* (MAs) need to be allocated inside a schedule to restore the maintenance level, preventing machine breakdown. Given that the machine can have a maximum maintenance level of $\mathrm{ML}^{\max}$, and assuming a unit maintenance speed, an MA of a duration D would increase the maintenance level by $\min\{D, \mathrm{ML}^{\max} - \mathrm{ML}\}$, where ML is the maintenance level before the MA.

With an initial machine maintenance level ML_0, $0 \leq \mathrm{ML}_0 \leq \mathrm{ML}^{\max}$, the goal of the problem is to schedule the jobs and necessary MAs such that all jobs can be processed without machine breakdown, and that the total completion time of jobs is minimized.

There are two variants of the problem depending on whether or not one has the freedom to choose the duration of an MA: in the *partial maintenance* case, the duration of each MA can be anywhere in between 0 and $(\mathrm{ML}^{\max} - \mathrm{ML})$, where ML is the maintenance level before the MA; in the *full maintenance* case, however, the duration of every MA must be exactly $(\mathrm{ML}^{\max} - \mathrm{ML})$, consequently increasing the maintenance level to the maximum value $\mathrm{ML}^{\max}$. Let C_i denote the completion time of the job J_i, for $i = 1, 2, \ldots, n$. In the three field notation, the two problems discussed in this paper are denoted as $(1|p\,\mathrm{MA}\,|\sum_i C_i)$ and $(1|f\,\mathrm{MA}\,|\sum_i C_i)$, respectively, where $p\,\mathrm{MA}$ and $f\,\mathrm{MA}$ refer to the partial and the full maintenance, respectively.

1.2 Prior work and our contribution

Bock *et al.* [1] proved that $(1|f\,\mathrm{MA}\,|\sum_i C_i)$ is NP-hard, even when $p_i = p$ for all i or when $p_i = \delta_i$ for all i, both by a reduction from the Partition problem [3]; while all the jobs have the same deterioration, i.e. $\delta_i = \delta$ for all i, the problem can be solved in $O(n \log n)$ time. For the partial maintenance case, Bock *et al.* [1] showed that the SPT rule gives an optimal schedule for $(1|p\,\mathrm{MA}\,|\sum_i C_i)$ when $p_i < p_j$ implies $p_i + \delta_i \leq p_j + \delta_j$ for each pair of i and j (which includes the special cases where $p_i = p$ for all i, or $\delta_i = \delta$ for all i, or $p_i = \delta_i$ for all i). The complexity of the general problem $(1|p\,\mathrm{MA}\,|\sum_i C_i)$ was left as an open problem. Also, to the best of our knowledge, no approximation algorithms have been designed for either problem.

Our main contribution in this paper is to settle the NP-hardness of the general problem $(1|p\,\mathrm{MA}\,|\sum_i C_i)$. Such an NP-hardness might appear a bit surprising at the first glance since one has so much freedom in choosing the starting time and the duration of each MA. Our reduction is from the PARTITION problem too, using a kind of job swapping argument. This reduction is presented in Section 3, following some preliminary properties we observe for the problem in Section 2. In Section 4, we propose a 2-approximation algorithm for $(1|p\,\mathrm{MA}\,|\sum_i C_i)$. We conclude the paper in Section 5 with some discussion on the (in-)approximability.

Lastly, we would like to point out that when the objective is to minimize the makespan $C_{\max}$, i.e. the maximum completion time of jobs, $(1|p\,\mathrm{MA}\,|C_{\max})$ can be trivially solved in $O(n)$ time and $(1|f\,\mathrm{MA}\,|C_{\max})$ is NP-hard but admits an $O\left(n^2(\mathrm{ML}^{\max})^2 \log\left(\sum_{i=1}^{n}(p_i+\delta_i)\right)\right)$ time algorithm based on dynamic programming (and thus admits an FPTAS) [1].

2 Preliminaries

Given a feasible schedule π to the problem $(1|p\,\mathrm{MA}\,|\sum_i C_i)$, which specifies the start processing time for each job and the starting time and the duration of each MA, we abuse slightly π to also denote the permutation of the job indices $(1, 2, \ldots, n)$ in which the jobs are processed in order: $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$. The following lemma is proved in [1].

▶ **Lemma 1** ([1]). *There is an optimal schedule π to $(1|p\,\mathrm{MA}\,|\sum_i C_i)$ such that the total maintenance duration before processing the job J_{π_i} equals $\max\left\{0, \sum_{j=1}^{i}\delta_{\pi_j} - \mathrm{ML}_0\right\}$, for each $i = 1, 2, \ldots, n$.*

Lemma 1 essentially states that each MA should be pushed later in the schedule as much as possible until absolutely necessary, and its duration should be minimized just for processing the succeeding job. In the sequel, we limit our discussion on the feasible schedules satisfying these two properties. We define the *separation job* in such a schedule π as the first job that requires an MA (of a positive duration).

▶ **Lemma 2.** *Suppose J_{π_k} is the separation job in an optimal schedule π to $(1|p\,\mathrm{MA}\,|\sum_i C_i)$. Then,*

- *the jobs before the separation job J_{π_k} are scheduled in the SPT order;*
- *the jobs after the separation job J_{π_k} are scheduled in the* shortest sum-of-processing-time-and-deterioration first *(SSF) order;*
- *the jobs adjacent to the separation job J_{π_k} satisfy*

$$p_{\pi_{k-1}} + \min\{\delta_{\pi_{k-1}}, \delta_{\pi_k} - \delta\} \le p_{\pi_k} + (\delta_{\pi_k} - \delta) \le p_{\pi_{k+1}} + \max\{0, \delta_{\pi_{k+1}} - \delta\},$$

where $\delta = \mathrm{ML}_0 - \sum_{i=1}^{k-1}\delta_{\pi_i}$ is the remaining maintenance level before the first MA.

Proof. Starting with an optimal schedule satisfying the properties stated in Lemma 1, one may apply a simple job swapping procedure if the job order is violated either in the prefix or in the suffix of job order separated by the separation job J_{π_k}. This procedure would decrease the value of the objective, contradicting to the optimality. That is, we have (see Figure 1 for an illustration)

$$p_{\pi_1} \le p_{\pi_2} \le \ldots \le p_{\pi_{k-1}}, \quad \text{and} \tag{1}$$

$$p_{\pi_{k+1}} + \delta_{\pi_{k+1}} \le p_{\pi_{k+2}} + \delta_{\pi_{k+2}} \le \ldots \le p_{\pi_n} + \delta_{\pi_n}. \tag{2}$$

Let $\delta = \mathrm{ML}_0 - \sum_{i=1}^{k-1}\delta_{\pi_i}$ denote the remaining maintenance level before the first MA. Because $\delta < \delta_{\pi_k}$, an (the first) MA of duration $\delta_{\pi_k} - \delta$ needs to be performed for processing

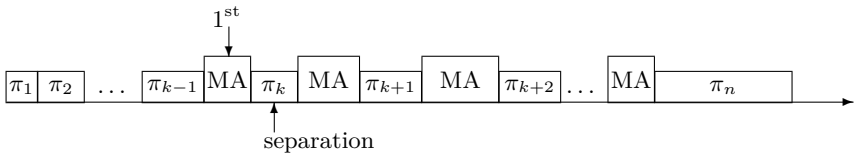

Figure 1 An illustration of the optimal schedule π stated in Lemma 2, where the separation job is J_{π_k}; the width of a framebox does not necessarily equal the processing time of a job or the duration of an MA.

the separation job J_{π_k}. From the optimality of π, swapping the two jobs J_{π_k} and $J_{\pi_{k+1}}$ should not decrease the objective, that is,

$$\begin{cases} p_{\pi_k} + (\delta_{\pi_k} - \delta) \le p_{\pi_{k+1}} + (\delta_{\pi_{k+1}} - \delta), & \text{if } \delta_{\pi_{k+1}} > \delta; \\ p_{\pi_k} + (\delta_{\pi_k} - \delta) \le p_{\pi_{k+1}}, & \text{otherwise.} \end{cases}$$

Similarly, swapping the two jobs $J_{\pi_{k-1}}$ and J_{π_k} should not decrease the objective, that is,

$$\begin{cases} p_{\pi_{k-1}} \le p_{\pi_k}, & \text{if } \delta_{\pi_{k-1}} \ge \delta_{\pi_k} - \delta; \\ p_{\pi_{k-1}} + \delta_{\pi_{k-1}} \le p_{\pi_k} + (\delta_{\pi_k} - \delta), & \text{otherwise.} \end{cases}$$

These together give

$$p_{\pi_{k-1}} + \min\{\delta_{\pi_{k-1}}, \delta_{\pi_k} - \delta\} \le p_{\pi_k} + (\delta_{\pi_k} - \delta) \le p_{\pi_{k+1}} + \max\{0, \delta_{\pi_{k+1}} - \delta\}. \tag{3}$$

This proves the lemma. ◀

From Lemma 2, one sees that the separation job in an optimal schedule is unique, in the sense that it cannot always be "appended" to either the prefix SPT order or the suffix SSF order. This is reflected in our NP-completeness reduction in Section 3, where we force a certain scenario to happen.

3 NP-hardness of the problem $(1|p\,\mathrm{MA}\,|\sum_i C_i)$

Our reduction is from the classic NP-complete problem Partition [3], formally defined as follows:

Partition
Instance: A set X of n positive integers $X = \{x_1, x_2, \ldots, x_n\}$, with $\sum_{i=1}^{n} x_i = 2B$.
Query: Is there a subset $X_1 \subset X$ such that $\sum_{x \in X_1} x = \sum_{x \in X - X_1} x = B$?

We abuse X to denote the instance of Partition with the set $X = \{x_1, x_2, \ldots, x_n\}$ and $\sum_{i=1}^{n} x_i = 2B$. The corresponding instance I of the problem $(1|p\,\mathrm{MA}\,|\sum_i C_i)$ is constructed in polynomial time, as follows:

Number of jobs:	$2n + 3$;
Job processing time:	$p_{n+1+i} = p_i = \sum_{j=1}^{i} x_j$, for $i = 0, 1, 2, \ldots, n$,
	$p_{2n+2} = M - 2B$;
Machine deterioration:	$\delta_{n+1+i} = \delta_i = M - 2p_i$, for $i = 0, 1, 2, \ldots, n$,
	$\delta_{2n+2} = 0$;
Initial maintenance level:	$\mathrm{ML}_0 = \sum_{i=0}^{n} \delta_i - 2B$;
Maximum maintenance level:	$\mathrm{ML}^{\max} = \sum_{i=0}^{n} \delta_i$;
Objective threshold:	$Q = Q_0 + B$,

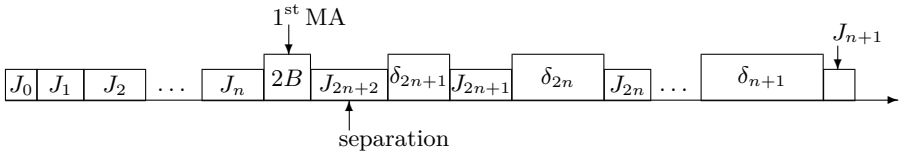

Figure 2 The initial infeasible schedule π^0 for the instance I with the separation job J_{2n+2}; π^0 satisfies all properties stated in Lemma 2. All MAs are indicated by their respective durations (for the first MA, its duration is $\delta_{2n+2} - \delta = 2B$).

(note that $p_{n+1} = p_0 = \sum_{j=1}^{0} x_j = 0$ due to the empty range for j) where M is a big integer:

$$M > (4n + 8)B, \tag{4}$$

and Q_0 is the total completion time of jobs for an *initial infeasible schedule* π^0 (see Figure 2):

$$Q_0 = \sum_{j=0}^{n}(n-j+1)p_j + (n+2)\left(\sum_{j=0}^{n} p_j + 2B + p_{2n+2}\right) + \sum_{j=0}^{n}(j+1)(p_{n+1+j} + \delta_{n+1+j}). \tag{5}$$

The job order in this initial schedule π^0 is $(J_0, J_1, \ldots, J_n, J_{2n+2}, J_{2n+1}, J_{2n}, \ldots, J_{n+1})$, and the first MA precedes the job J_{2n+2}, which is regarded as the separation job (see Figure 2). Before the separation job J_{2n+2}, the machine maintenance level is allowed to go into negative, but has to be restored to zero just for processing J_{2n+2}; afterwards, machine breakdown is no longer tolerated. From $\mathrm{ML}_0 = \sum_{i=0}^{n} \delta_i - 2B$, we know that π^0 is *infeasible* due to machine breakdown before the first MA; we will convert it to a feasible schedule later. The QUERY of the decision version of the problem $(1|p\,\mathrm{MA}\,|\sum_i C_i)$ is whether or not there exists a feasible schedule π such that the total completion time of jobs is no more than $Q = Q_0 + B$.

Despite the infeasibility, the initial schedule π^0 has all the properties stated in Lemma 2, with the separation job J_{2n+2} at the *center position*. The first $(n+1)$ jobs are in the SPT order and the last $(n+1)$ jobs are in the SSF order; since $\delta = -2B$, $p_n = p_{2n+1} = 2B$, $\delta_n = \delta_{2n+1} = M - 4B$, $p_{2n+2} = M - 2B$, $\delta_{2n+2} = 0$, Eq. (3) is also satisfied due to the big M in Eq. (4):

$$p_n + \min\{\delta_n, \delta_{2n+2} - \delta\} < p_{2n+2} + (\delta_{2n+2} - \delta) = p_{2n+1} + \max\{0, \delta_{2n+1} - \delta\}.$$

In the rest of the section, we will show that there is a subset $X_1 \subset X$ of sum exactly B if and only if the initial schedule π^0 can be converted into a feasible schedule π with the total completion time of jobs no more than $Q = Q_0 + B$, through a *repeated job swapping* procedure.

Notice that the two jobs J_i and J_{n+1+i} are identical, for $i = 0, 1, \ldots, n$. In any schedule with the job J_{2n+2} at the center position, if exactly one of J_i and J_{n+1+i} is scheduled before J_{2n+2}, then we always say J_i is scheduled before J_{2n+2} while J_{n+1+i} is scheduled after J_{2n+2}. Also, when the two jobs J_i and J_{n+1+i} are both scheduled before J_{2n+2}, then J_{n+1+i} precedes J_i; when the two jobs J_i and J_{n+1+i} are both scheduled after J_{2n+2}, then J_i precedes J_{n+1+i}.

3.1 Proof of "only if"

In this subsection, we show that if there is a subset $X_1 \subset X$ of sum exactly B, then the initial infeasible schedule π^0 can be converted into a feasible schedule π with the total completion

time no more than $Q = Q_0 + B$. We also demonstrate the repeated job swapping procedure leading to this successful schedule π.

Suppose the indices of the elements in the subset X_1 are $\{i_1, i_2, \ldots, i_m\}$, satisfying $1 \le i_1 < i_2 < \ldots < i_m \le n$. Starting with the initial schedule π^0, we sequentially swap the job $J_{i_\ell - 1}$ with the job J_{n+1+i_ℓ}, for $\ell = 1, 2, \ldots, m$. Let π^ℓ denote the schedule after the ℓ-th job swapping.

▶ **Lemma 3.** *For each $1 \le \ell \le m$,*

- *the schedule π^ℓ with the separation job J_{2n+2} satisfies the properties in Lemma 2;*
- *the ℓ-th job swapping decreases the total machine deterioration before the separation job J_{2n+2} by $2x_{i_\ell}$;*
- *the ℓ-th job swapping increases the total completion time by x_{i_ℓ}.*

Proof. Recall that the two jobs J_{i_ℓ} and J_{n+1+i_ℓ} are identical. Before the ℓ-th job swapping between $J_{i_\ell - 1}$ and J_{n+1+i_ℓ} (in the schedule $\pi^{\ell-1}$), the jobs in between $J_{i_\ell - 1}$ and J_{n+1+i_ℓ} are

$$(J_{i_\ell - 1}, J_{i_\ell}, J_{i_\ell + 1}, \ldots, J_n, J_{2n+2}, J_{2n+1}, J_{2n}, \ldots, J_{n+1+i_\ell+1}, J_{n+1+i_\ell}).$$

After the swapping (in the schedule π^ℓ) this sub-schedule becomes

$$(J_{n+1+i_\ell}, J_{i_\ell}, J_{i_\ell + 1}, \ldots, J_n, J_{2n+2}, J_{2n+1}, J_{2n}, \ldots, J_{n+1+i_\ell+1}, J_{i_\ell - 1}).$$

By a simple induction, all jobs before J_{n+1+i_ℓ} have their processing times less than p_{i_ℓ}, and thus the jobs before the separation job J_{2n+2} are in the SPT order; for a similar reason, the jobs after the separation job J_{2n+2} are in the SSF order.

By the ℓ-th job swapping, the change in the total machine deterioration before the separation job J_{2n+2} is $\delta_{i_\ell} - \delta_{i_\ell - 1} = -2(p_{i_\ell} - p_{i_\ell - 1}) = -2x_{i_\ell}$, that is, decreases by $2x_{i_\ell}$. Therefore the duration of the first MA also decreases by $2x_{i_\ell}$. Since J_n always directly precedes J_{2n+2} and $p_n < p_{2n+2}$, the first half of Eq. (3) holds; since $p_{2n+2} + \delta_{2n+2}$ is the smallest among all jobs, the second half of Eq. (3) holds. That is, the schedule π^ℓ satisfies all properties in Lemma 2.

For ease of presentation, let C_i denote the completion time of the job J_i in the schedule π^ℓ, and let C'_i denote the completion time of the job J_i in the schedule $\pi^{\ell-1}$. Comparing to the schedule $\pi^{\ell-1}$ ($\ell \ge 1$), after the ℓ-th job swapping between $J_{i_\ell - 1}$ and J_{n+1+i_ℓ},

- the completion time of jobs preceding J_{n+1+i_ℓ} is unchanged;
- $C_{n+1+i_\ell} - C'_{i_\ell - 1} = p_{i_\ell} - p_{i_\ell - 1} = x_{i_\ell}$;
- the completion time of each job in between J_{i_ℓ} and J_n (inclusive, $n - i_\ell + 1$ of them) increases by x_{i_ℓ};
- the duration of the first MA decreases by $2x_{i_\ell}$;
- the completion time of each job in between J_{2n+2} and $J_{n+1+i_\ell+1}$ (inclusive, $n - i_\ell + 1$ of them) decreases by x_{i_ℓ};
- $C_{i_\ell - 1} - C'_{n+1+i_\ell} = -x_{i_\ell} + (\delta_{i_\ell - 1} + p_{i_\ell - 1}) - (\delta_{i_\ell} + p_{i_\ell}) = 0$;
- from the last item, the completion time of jobs succeeding $J_{i_\ell - 1}$ is unchanged.

In summary, there are $(n - i_\ell + 2)$ jobs of which the completion time increases by x_{i_ℓ} and $(n - i_\ell + 1)$ jobs of which the completion time decreases by x_{i_ℓ}. Therefore, the ℓ-th job swapping between $J_{i_\ell - 1}$ and J_{n+1+i_ℓ} increases the total completion time by x_{i_ℓ}. This finishes the proof. ◀

▶ **Theorem 4.** *If there is a subset $X_1 \subset X$ of sum exactly B, then there is a feasible schedule π to the instance I with the total completion time no more than $Q = Q_0 + B$.*

Proof. Let the indices of the elements in the subset X_1 be $\{i_1, i_2, \ldots, i_m\}$, such that $1 \leq i_1 < i_2 < \ldots < i_m \leq n$. Starting with the initial schedule π^0, we sequentially swap the job $J_{i_\ell - 1}$ with the job J_{n+1+i_ℓ}, for $\ell = 1, 2, \ldots, m$. Let π^ℓ denote the schedule after the ℓ-th job swapping, and let Q_ℓ denote the total completion time of jobs in π^ℓ.

From Lemma 3 we know that the ending schedule π^m satisfies all the properties in Lemma 2. Also, the total machine deterioration before the separation job J_{2n+2} in π^m is

$$\sum_{i=0}^{n} \delta_i - 2\sum_{\ell=1}^{m} x_{i_\ell} = \sum_{i=0}^{n} \delta_i - 2B = \mathrm{ML}_0,$$

suggesting that π^m is a feasible schedule. (The first MA has zero duration and thus becomes unnecessary.)

Moreover, the total completion time of jobs in π^m is $Q_m = Q_0 + \sum_{\ell=1}^{m} x_{i_\ell} = Q_0 + B$. Therefore, the schedule π^m obtained from the initial schedule π^0 through the repeated job swapping procedure is a desired one. ◀

3.2 Proof of "if"

In this subsection, we show that if there is a feasible schedule π to the constructed instance I with the total completion time no more than $Q = Q_0 + B$, then there is a subset $X_1 \subset X$ of sum exactly B. Assume without loss of generality that the schedule π satisfies the properties in Lemma 2. We start with some structure properties which the schedule π must have; the interested readers may refer to [9] for detailed proofs.

▶ **Lemma 5** ([9]). *Excluding the job J_{2n+2}, there are at least n and at most $(n+1)$ jobs scheduled before the first* MA *in the schedule π.*

▶ **Lemma 6** ([9]). *There are at most $(n+1)$ jobs scheduled after J_{2n+2} in the schedule π.*

Combining Lemmas 5 and 6, we have the following lemma regarding the position of J_{2n+2} in the schedule π.

▶ **Lemma 7** ([9]). *In the schedule π, the position of the job J_{2n+2} has three possibilities:*

Case 1: *There are $(n+2)$ jobs before the first* MA *with $\pi_{n+2} = 2n+2$, and $J_{\pi_{n+3}}$ is the separation job.*

Case 2: *There are $(n+1)$ jobs before the first* MA*, $J_{\pi_{n+2}}$ is the separation job, and $\pi_{n+3} = 2n+2$.*

Case 3: *There are n jobs before the first* MA*, $J_{\pi_{n+1}}$ is the separation job, and $\pi_{n+2} = 2n+2$.*

Recall that the job order in the initial infeasible schedule π^0 is $(J_0, J_1, \ldots, J_n, J_{2n+2}, J_{2n+1}, J_{2n}, \ldots, J_{n+2}, J_{n+1})$, and the first MA is executed before processing the job J_{2n+2}, which is regarded as the separation job (see Figure 2). In the sequel, we will again convert π^0 into our target schedule π through a repeated job swapping procedure. During such a procedure, the job J_{2n+2} is kept at the center position, and a job swapping always involves a job before J_{2n+2} and a job after J_{2n+2}.

In Cases 1 and 3 of the schedule π, the job J_{2n+2} is at the center position (recall that there are in total $2n+3$ jobs), and therefore the target schedule is well set. In Case 2, J_{2n+2} is at position $n+3$, not the center position; we first exchange J_{2n+2} and $J_{\pi_{n+2}}$ to obtain a schedule π', which becomes our target schedule. That is, we will first convert π^0 into π' through a repeated job swapping procedure, and at the end exchange J_{2n+2} back to the position $n+3$ to obtain the final schedule π. In summary, our primary goal is to convert

the schedule π^0 through a repeated job swapping procedure, keeping the job J_{2n+2} at the center position and keeping the first MA right before the job J_{2n+2} (to be detailed next). At the end, to obtain the target schedule π, in Case 1, we swap the job J_{2n+2} and the first MA (*i.e.*, moving the first MA one position backward); in Case 2, we swap J_{2n+2} and the immediate succeeding MA and the following job (with the MA merged with the first MA); in Case 3, we swap the first MA and its immediate preceding job (*i.e.*, moving the first MA one position forward).

In the target schedule (π in Cases 1 and 3, or π' in Case 2), let $R = \{r_1, r_2, \ldots, r_m\}$ denote the subset of indices such that both J_{r_j} and J_{n+1+r_j} are among the first $(n+1)$ jobs, where $0 \le r_1 < r_2 < \ldots < r_m \le n$, and $L = \{\ell_1, \ell_2, \ldots, \ell_m\}$ denote the subset of indices such that both J_{ℓ_j} and $J_{n+1+\ell_j}$ are among the last $(n+1)$ jobs, where $0 \le \ell_1 < \ell_2 < \ldots < \ell_m \le n$. Note that J_{2n+2} is at the center position in the target schedule, and thus it has to be $|R| = |L|$ and we let $m = |R|$. Clearly, all these ℓ_j's and r_j's are distinct from each other.

In the repeated job swapping procedure leading the initial infeasible schedule π^0 to the target feasible schedule, the j-th job swapping is to swap the two jobs J_{ℓ_j} and J_{n+1+r_j}. The resultant schedule after the j-th job swapping is denoted as π^j, for $j = 1, 2, \ldots, m$. In Section 3.1, the job swapping is "*regular*" in the sense that $\ell_j = r_j - 1$ for all j, but now ℓ_j and r_j do not necessarily relate to each other. We remark that immediately after the swapping, a job sorting is needed to restore the SPT order for the prefix and the SSF order for the suffix (see the last paragraph before Section 3.1 for possible re-indexing the jobs).

The following Lemma 8 on the j-th job swapping, when $\ell_j < r_j$, is an extension of Lemma 3.

▶ **Lemma 8** ([9]). *For each $1 \le j \le m$, if the schedule π^{j-1} satisfies the first two properties in Lemma 2 and $\ell_j < r_j$, then*

- *the schedule π^j satisfies the first two properties in Lemma 2;*
- *the j-th job swapping decreases the total machine deterioration before the center job J_{2n+2} by $\delta_{\ell_j} - \delta_{r_j} = 2\sum_{k=\ell_j+1}^{r_j} x_k$;*
- *the j-th job swapping increases the total completion time by at least $\sum_{k=\ell_j+1}^{r_j} x_k$; and the increment equals $\sum_{k=\ell_j+1}^{r_j} x_k$ if and only if $\ell_j > r_{j-1}$.*

▶ **Lemma 9** ([9]). *For each $1 \le j \le m$, if the schedule π^{j-1} satisfies the first two properties in Lemma 2 and $\ell_j > r_j$, then*

- *the schedule π^j satisfies the first two properties in Lemma 2;*
- *the j-th job swapping increases the total machine deterioration before the center job J_{2n+2} by $\delta_{r_j} - \delta_{\ell_j} = 2\sum_{k=r_j+1}^{\ell_j} x_k$;*
- *the j-th job swapping increases the total completion time by at least $\sum_{k=r_j+1}^{\ell_j} x_k$.*

▶ **Theorem 10.** *If there is a feasible schedule π to the instance I with the total completion time no more than $Q = Q_0 + B$, then there is a subset $X_1 \subset X$ of sum exactly B.*

Proof. We start with a feasible schedule π, which has the first two properties stated in Lemma 2 and for which the total completion time is no more than $Q = Q_0 + B$. Excluding the job J_{2n+2}, using the first $n+1$ jobs and the last $n+1$ job in π, we determine the two subsets of indices $R = \{r_1, r_2, \ldots, r_m\}$ and $L = \{\ell_1, \ell_2, \ldots, \ell_m\}$, and define the corresponding m job swappings. We then repeatedly apply the job swapping to convert the initial infeasible schedule π^0 into π.

In Case 1, the total machine deterioration of the first $(n+1)$ jobs in π is

$$\sum_{i=0}^{n} \delta_i - 2\sum_{\ell_j < r_j} \sum_{k=\ell_j+1}^{r_j} x_k + 2\sum_{\ell_j > r_j} \sum_{k=r_j+1}^{\ell_j} x_k = \mathrm{ML}_0 - \delta,$$

implying that

$$\sum_{\ell_j<r_j}\sum_{k=\ell_j+1}^{r_j} x_k - \sum_{\ell_j>r_j}\sum_{k=r_j+1}^{\ell_j} x_k = B + \frac{1}{2}\delta, \tag{6}$$

where $\delta \geq 0$ is the remaining machine maintenance level before the first MA.

On the other hand, the total completion time of jobs in the schedule π is at least

$$Q_0 + \sum_{\ell_j<r_j}\sum_{k=\ell_j+1}^{r_j} x_k + \sum_{\ell_j>r_j}\sum_{k=r_j+1}^{\ell_j} x_k = Q_0 + B + \frac{1}{2}\delta + 2\sum_{\ell_j>r_j}\sum_{k=r_j+1}^{\ell_j} x_k.$$

It follows that 1) $\delta = 0$; 2) there is no pair of swapping jobs J_{ℓ_j} and J_{n+1+r_j} such that $\ell_j > r_j$; and 3) $\ell_1 < r_1 < \ell_2 < r_2 < \ldots < \ell_m < r_m$ (from the third item of Lemma 8). Therefore, from Eq. (6), for the subset $X_1 = \cup_{j=1}^{m}\{x_{\ell_j+1}, x_{\ell_j+2}, \ldots, x_{r_j}\}$, $\sum_{x\in X_1} x = B$. That is, the instance X of the PARTITION problem is a yes-instance.

In the other two cases, one can similarly, though a bit more complex, show that the instance X is a yes-instance. The detailed proofs are in [9]. ◀

The following theorem follows immediately from Theorems 4 and 10.

▶ **Theorem 11.** *The general problem* $(1|p\,\mathrm{MA}\,|\sum_j C_j)$ *is NP-hard.*

4 A 2-approximation algorithm for $(1|p\,\mathrm{MA}\,|\sum_j C_j)$

Recall that in the problem $(1|p\,\mathrm{MA}\,|\sum_j C_j)$, we are given a set of jobs $\mathcal{J} = \{J_i, i = 1, 2, \ldots, n\}$, where each job $J_i = (p_i, \delta_i)$ is specified by its non-preemptive *processing time* p_i and *machine deterioration* δ_i. The machine deterioration δ_i quantifies the decrement in the machine maintenance level after processing the job J_i. The machine has an initial machine maintenance level ML_0, $0 \leq \mathrm{ML}_0 \leq \mathrm{ML}^{\max}$, where $\mathrm{ML}^{\max}$ is the maximum maintenance level. The goal is to schedule the jobs and necessary MAs of any duration such that all jobs can be processed without machine breakdown, and that the total completion time of jobs is minimized.

In this section, we present a 2-approximation algorithm, denoted as $\mathcal{A}_1$, for the problem. Furthermore, the algorithm $\mathcal{A}_1$ produces a feasible schedule π satisfying the first two properties stated in Lemma 2, suggesting that if the third property is violated then a local job swapping can further decrease the total completion time.

In the algorithm $\mathcal{A}_1$, the first step is to sort the jobs in SSF order (and thus we assume without loss of generality that) $p_1 + \delta_1 \leq p_2 + \delta_2 \leq \ldots \leq p_n + \delta_n$. In the second step, the separation job is determined to be J_k, where k is the maximum index such that $\sum_{i=1}^{k-1}\delta_i \leq \mathrm{ML}_0$. In the last step, the jobs preceding the separation job J_k are re-sorted in the SPT order, denoted by $(J_{i_1}, J_{i_2}, \ldots, J_{i_{k-1}})$, and the jobs succeeding the separation job are $(J_{k+1}, J_{k+2}, \ldots, J_n)$. That is, the solution schedule is

$$\pi = (J_{i_1}, J_{i_2}, \ldots, J_{i_{k-1}}; \mathrm{MA}_1, J_k; \mathrm{MA}_2, J_{k+1}, \mathrm{MA}_3, J_{k+2}, \ldots, \mathrm{MA}_{n-k+1}, J_n),$$

where $\mathrm{MA}_1 = \sum_{j=1}^{k}\delta_j - \mathrm{ML}_0$ and $\mathrm{MA}_i = \delta_{k-1+i}$ for $i = 2, 3, \ldots, n-k+1$.

Let π^* denote an optimal schedule satisfying all properties stated in Lemma 2, and its separation job is $J_{\pi^*_{k^*}}$:

$$\pi^* = (J_{\pi^*_1}, J_{\pi^*_2}, \ldots, J_{\pi^*_{k^*-1}}; \mathrm{MA}^*_1, J_{\pi^*_{k^*}}; \mathrm{MA}^*_2, J_{\pi^*_{k^*+1}}, \mathrm{MA}^*_3, J_{\pi^*_{k^*+2}}, \ldots, \mathrm{MA}^*_{n-k^*+1}, J_{\pi^*_n}).$$

Let C_i (C_i^*, respectively) denote the completion time of the job J_{π_i} ($J_{\pi_i^*}$, respectively) in the schedule π (π^*, respectively); the makespans of π and π^* are $C_{\max}$ and $C_{\max}^*$, respectively, and (recall that $\mathrm{ML}_0 < \sum_{i=1}^{n} \delta_i$)

$$C_{\max} = C_{\max}^* = \sum_{i=1}^{n}(p_i + \delta_i) - \mathrm{ML}_0\,. \tag{7}$$

▶ **Lemma 12.** *For every $i \geq k$ we have*

$$\sum_{j=i}^{n}(p_j + \delta_j) \geq \sum_{j=i}^{n}(p_{\pi_j^*} + \delta_{\pi_j^*}).$$

Proof. Since $p_1 + \delta_1 \leq p_2 + \delta_2 \leq \ldots \leq p_n + \delta_n$, $\sum_{j=i}^{n}(p_j + \delta_j)$ is the maximum sum of processing times and machine deterioration, over all possible subsets of $(n - i + 1)$ jobs. The lemma thus holds. ◀

▶ **Theorem 13.** *The algorithm $\mathcal{A}_1$ is an $O(n \log n)$-time 2-approximation algorithm for the problem $(1|p\,\mathrm{MA}\,|\sum_j C_j)$.*

Proof. We compare the two schedules π obtained by the algorithm $\mathcal{A}_1$ and π^* an optimal schedule satisfying the properties stated in Lemma 2. Using Eq. (7) and Lemma 12, it is clear that $C_i \leq C_i^*$ for each $i = n, n-1, \ldots, \max\{k, k^*\}$.

Suppose $k < k^*$, then for each i such that $k \leq i < k^*$, we have

$$\begin{array}{rclcl} C_i & = & C_n - \sum_{j=i+1}^{n}(p_j + \delta_j) & \leq & C_n^* - \sum_{j=i+1}^{n}(p_{\pi_j^*} + \delta_{\pi_j^*}) \\ & = & \sum_{j=1}^{i}(p_{\pi_j^*} + \delta_{\pi_j^*}) - \mathrm{ML}_0 & = & \sum_{j=1}^{i} p_{\pi_j^*} - \left(\mathrm{ML}_0 - \sum_{j=1}^{i}\delta_{\pi_j^*}\right) \\ & \leq & \sum_{j=1}^{i} p_{\pi_j^*} & = & C_i^*. \end{array}$$

Therefore, we have $C_i \leq C_i^*$ for each $i = n, n-1, \ldots, k$. It follows that

$$\sum_{i=k}^{n} C_i \leq \sum_{i=k}^{n} C_i^* \leq \mathrm{OPT}. \tag{8}$$

On the other hand, by the SPT order, the algorithm $\mathcal{A}_1$ achieves the minimum total completion time of jobs of $\{J_1, J_2, \ldots, J_{k-1}\}$. One clearly sees that in the optimal schedule π^*, the sub-total completion time of $\{J_1, J_2, \ldots, J_{k-1}\}$ is upper-bounded by OPT. Therefore,

$$\sum_{i=1}^{k-1} C_i \leq \mathrm{OPT}. \tag{9}$$

Merging Eqs. (8) and (9), we conclude that the total completion time of schedule π is

$$\sum_{i=1}^{k-1} C_i + \sum_{i=k}^{n} C_i \leq 2 \cdot \mathrm{OPT}.$$

This proves the performance ratio of 2 (which can also be shown tight on a trivial 2-job instance $I = \{J_1 = (1, \lambda), J_2 = (\lambda - 1, 1), \mathrm{ML}_0 = \mathrm{ML}^{\max} = \lambda\}$, with a large λ). The running time of the algorithm $\mathcal{A}_1$ is dominated by two times of sorting, each takes $O(n \log n)$ time. ◀

5 Concluding remarks

We investigated the single machine scheduling with job-dependent machine deterioration, recently introduced by Bock *et al.* [1], with the objective to minimize the total completion time of jobs. In the partial maintenance case, we proved the NP-hardness for the general problem, thus addressing the open problem left in the previous work. From the approximation perspective, we designed a 2-approximation, for which the ratio 2 is tight on a trivial two-job instance.

The 2-approximation algorithm is simple, but it is the first such work. Our major contribution is the non-trivial NP-hardness proof, which might appear surprising at the first glance since one has so much freedom in choosing the starting time and the duration of the maintenance activities. It would be interesting to further study the (in-)approximability for the problem. It would also be interesting to study the problem in the full maintenance case, which was shown NP-hard, from the approximation algorithm perspective. Approximating the problem in the full maintenance case seems more challenging, where we need to deal with multiple bin-packing sub-problems, while the inter-relationship among them is much complex.

References

1 S. Bock, D. Briskorn, and A. Horbach. Scheduling flexible maintenance activities subject to job-dependent machine deterioration. *Journal of Scheduling*, 15:565–578, 2012.

2 J.-S. Chen. Scheduling of non-resumable jobs and flexible maintenance activities on a single machine to minimize makespan. *European Journal of Operation Research*, 190:90–102, 2008.

3 M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. W. H. Freeman and Company, San Francisco, 1979.

4 M. A. Kubzin and V. A. Strusevich. Planning machine maintenance in two-machine shop scheduling. *Operations Research*, 54:789–800, 2006.

5 C.-Y. Lee. Machine scheduling with availability constraints. In J. Y.-T. Leung, editor, *Handbook of Scheduling: Algorithms, Models and Performance Analysis*, pages 22: 1–13. 2004.

6 C.-Y. Lee and Z.-L. Chen. Scheduling jobs and maintenance activities on parallel machines. *Naval Research Logistics*, 47:145–165, 2000.

7 C.-Y. Lee and S. Liman. Single machine flow-time scheduling with scheduled maintenance. *Acta Informatica*, 29:375–382, 1992.

8 W. Luo, L. Chen, and G. Zhang. Approximation algorithms for scheduling with a variable machine maintenance. In *Proceedings of Algorithmic Aspects in Information and Management (AAIM 2010)*, LNCS 6124, pages 209–219, 2010.

9 W. Luo, Y. Xu, W. Tong, and G. Lin. Single machine scheduling with job-dependent machine deterioration. *CoRR*, abs/1606.04157, 2016.

10 Y. Ma, C. Chu, and C. Zuo. A survey of scheduling with deterministic machine availability constraints. *Computers & Industrial Engineering*, 58:199–211, 2010.

11 G. Mosheiov and A. Sarig. Scheduling a maintenance activity to minimize total weighted completion-time. *Computers & Mathematics with Applications*, 57:619–623, 2009.

12 X. Qi. A note on worst-case performance of heuristics for maintenance scheduling problems. *Discrete Applied Mathematics*, 155:416–422, 2007.

13 X. Qi, T. Chen, and F. Tu. Scheduling the maintenance on a single machine. *Journal of the Operational Research Society*, 50:1071–1078, 1999.

14 G. Schmidt. Scheduling with limited machine availability. *European Journal of Operational Research*, 121:1–15, 2000.

15 K. Sun and H. Li. Scheduling problems with multiple maintenance activities and non-preemptive jobs on two identical parallel machines. *International Journal of Production Economics*, 124:151–158, 2010.

16 D. Xu, Y. Yin, and H. Li. Scheduling jobs under increasing linear machine maintenance time. *Journal of Scheduling*, 13:443–449, 2010.

17 S. Yang and D. Yang. Minimizing the makespan on single-machine scheduling with aging effect and variable maintenance activities. *Omega*, 38:528–533, 2010.