mathematics

MDPI

Article

Online Batch Scheduling of Simple Linear Deteriorating Jobs with Incompatible Families

Wenhua Li [1,*], Libo Wang [1], Xing Chai [2] and Hang Yuan [3]

[1] School of Mathematics and Statistics, Zhengzhou University, Zhengzhou 450001, China; wanglibo0224@163.com
[2] College of Science, Henan University of Technology, Zhengzhou 450001, China; chaixingstudy@163.com
[3] Department of Economics, State University of New York at Binghamton, Binghamton, NY 13902, USA; hyuan3@binghamton.edu
* Correspondence: liwenhua@zzu.edu.cn

Received: 31 December 2019; Accepted: 24 January 2020; Published: 1 February 2020

Abstract: We considered the online scheduling problem of simple linear deteriorating job families on m parallel batch machines to minimize the makespan, where the batch capacity is unbounded. In this paper, simple linear deteriorating jobs mean that the actual processing time p_j of job J_j is assumed to be a linear function of its starting time s_j, i.e., $p_j = \alpha_j s_j$, where $\alpha_j > 0$ is the deterioration rate. Job families mean that one job must belong to some job family, and jobs of different families cannot be processed in the same batch. When $m = 1$, we provide the best possible online algorithm with the competitive ratio of $(1 + \alpha_{\max})^f$, where f is the number of job families and $\alpha_{\max}$ is the maximum deterioration rate of all jobs. When $m \geq 1$ and $m = f$, we provide the best possible online algorithm with the competitive ratio of $1 + \alpha_{\max}$.

Keywords: online algorithm; batch scheduling; linear deterioration; job families; competitive ratio

1. Introduction

1.1. Background

In this paper, all jobs arrive over time, i.e., each job has an arrival time. Before the jobs arrive, we do not know any information, including arrival time, processing time, deterioration rate, etc. Due to the unknown information of the jobs, the online algorithm is not guaranteed to be optimal. Borodin and El-Yaniv [1] used the competitive ratio to measure the quality of an online algorithm. For a minimization scheduling problem, we define the competitive ratio of the online algorithm A as:

$$\rho = \sup\{A(\mathcal{I})/OPT(\mathcal{I}) : \mathcal{I} \text{ is any instance such that } OPT(\mathcal{I}) > 0\}.$$

where $\mathcal{I}$ is any job instance and $A(\mathcal{I})$ and $OPT(\mathcal{I})$ are the objective values obtained from the algorithm A and an optimal offline scheduling OPT, respectively. In this study, the objective was to minimize the makespan. An online algorithm A is called the best possible if no other online algorithms A^* produce a smaller competitive ratio.

Parallel-batch means that one batch processing machine can process b jobs simultaneously as a batch. The processing time of a batch is the maximum processing time of all jobs in this batch. All jobs in a batch have the same starting time, processing time, and completion time. According to the number of jobs contained in a batch, Brucker et al. [2] divided the model into two cases: the unbounded model ($b = \infty$) and the bounded model ($b < \infty$).

Job families mean that one job must belong to some job family, and jobs of different families cannot be processed in the same batch. Online scheduling problems on parallel batch machines with

incompatible job families have been studied extensively. Fu et al. [3] studied the online algorithm on a single machine to minimize the makespan. Li et al. [4] examined the online scheduling of incompatible unit-length job families with lookahead on a single machine. Tian et al. [5] analyzed the problem on m parallel machines. However, research is lacking on the parallel-batch online scheduling with incompatible deteriorating job families.

Traditional scheduling problems assume that the processing time of a job is fixed. However, in real life, one job will take longer when it has a later starting time. For example, in steel production and financial management [6,7], the processing time is longer when it starts later. In the steel-making process, strict requirements are placed on temperature. If the waiting time is too long, the temperature of molten steel will drop. So, it will take time to heat up again before further processing. Other examples are provided in cleaning and fire fighting. The scheduling problem of deteriorating jobs was first introduced by Browne and Yechiali [8] and Gupta and Gupta [9], independently. Both considered minimizing makespan on a single machine. Since then, this topic has attracted considerable attentions. Gawiejnowicz and Kononov [10] considered the general properties of scheduling with fixed job processing time and scheduling with job processing time as proportional linear functions of the job starting time. The relevant research includes [11–18], among many others. Recently, some works have been published about online algorithms for linear deteriorating jobs [19–23].

To minimize the makespan of the online scheduling problem on m parallel machines with linear deteriorating jobs, Cheng et al. [19] constructed an algorithm and proved that the bound of the competitive ratio of the algorithm is tight, where $m = 2$ and the largest deterioration rate of jobs is known in advance. Yu et al. [22] proved that no deterministic online algorithm is better than $(1 + \alpha_{\max})$-competitive when $m = 2$, where $\alpha_{\max}$ is the maximum deterioration rate of all jobs.

1.2. Research Problem

Our contribution is to extend the online scheduling problem on m parallel batch machines with simple linear deteriorating job families to minimize the makespan. Here, batch capacity is unbounded, i.e., $b = \infty$. We use f-family to denote there are f job families. We constructed the best possible online algorithm with the competitive ratio of $(1 + \alpha_{\max})^f$ when $m = 1$, where f is the number of job families and $\alpha_{\max}$ is the maximum deterioration rate of all jobs. When $m \geq 1$ and $m = f$, we created the best possible online algorithm with the competitive ratio of $1 + \alpha_{\max}$.

We examined the online batch scheduling of simple linear deteriorating job families. The actual processing time p_j of job J_j is assumed to be a linear function of its starting time s_j, i.e., $p_j = \alpha_j s_j$, where $\alpha_j > 0$ is the deterioration rate, which is unknown until it arrives. The objective was to minimize makespan. Assume that the arrival time of all jobs is greater than or equal to $t_0 > 0$; otherwise, jobs arriving at time 0 can be completed at time 0. We used the three-field notation $\alpha|\beta|\gamma$ [24] to represent one scheduling problem.

This paper is organized as follows. In Section 2, we consider the problem 1|online,r_j,p-batch, $b = \infty$,f-family, $p_j = \alpha_j t|C_{\max}$, where f is the number of job families. We prove the lower bound and provide the best possible online algorithm with the competitive ratio of $(1 + \alpha_{\max})^f$. In Section 3, we consider the problem Pm|online,r_j,p-batch, $b = \infty$,m-family, $p_j = \alpha_j t|C_{\max}$, where m is the number of machines. We prove the lower bound and provide the best possible online algorithm with the competitive ratio of $1 + \alpha_{\max}$, where $\alpha_{\max}$ is the maximum deterioration rate of all jobs.

Throughout this paper, we use σ and π to denote the schedules obtained from an online algorithm and an optimal offline schedule, respectively. Let $C_{\max}(\sigma)$ and $C_{\max}(\pi)$ be the objective values of σ and π, respectively, and $\alpha_{\max}$ be the maximum deterioration rate of all jobs. Let ϵ be an arbitrary small positive number.

2. Single Batch Machine ($m = 1$)

In this section, we consider the online scheduling on an unbounded batch machine and the jobs belong to f incompatible deteriorating job families. The number of job families, f, is known in advance.

We prove the lower bound and provide the best possible online algorithm with the competitive ratio of $(1+\alpha_{\max})^f$.

Theorem 1. *For problem* $1|online,r_j,p\text{-}batch,b=\infty,f\text{-}family,p_j=\alpha_j t|C_{\max}$, *the competitive ratio of any online algorithm is not less than* $(1+\alpha_{\max})^f$.

Proof. Let H be any online algorithm and I be a job instance provided by the adversary. In instance I, all the jobs have the same deterioration rate of α.

At time t_0, f jobs from the f different job families arrive by the adversary. If a job is scheduled by H to process at time t and t is in the time interval $[t_0,(1+\alpha)^f t_0)$, then at time $t+\epsilon$, the adversary releases a copy of this job, which belongs to the same job family. Let s be the starting time of the first job whose completion time is at least $(1+\alpha)^f t_0$. $(1+\alpha)s \geq (1+\alpha)^f t_0$. so,

$$s \geq (1+\alpha)^{f-1} t_0. \tag{1}$$

Case 1 $(1+\alpha)^{f-1}t_0 \leq s < (1+\alpha)^f t_0$.

In this case, there are still f jobs from f distinct job families that are not processed at time $(1+\alpha)s$. So,

$$C_{\max}(\sigma) \geq (1+\alpha)^f (1+\alpha)s = (1+\alpha)^{f+1}s. \tag{2}$$

We assume that the jobs processed in the time interval $[t_0,(1+\alpha)s)$ belong to k distinct job families, say, $\mathcal{F}_1,\mathcal{F}_2,\cdots,\mathcal{F}_k$, where $1 \leq k \leq f$. The other $f-k$ job families are defined by $\mathcal{F}_{k+1},\mathcal{F}_{k+2},\cdots,\mathcal{F}_f$. Let s_i be the last starting time of the jobs in $\mathcal{F}_i$ that start before or at time s for $1 \leq i \leq k$, and satisfy $s_1 < s_2 < \cdots < s_k$. Clearly, $s_k = s$. From the construction of instance I, we know that the last arrival time of the jobs in $\mathcal{F}_i$ $(1 \leq i \leq k)$ is $s_i+\epsilon$ and the arrival time of the jobs in $\mathcal{F}_i$ $(k+1 \leq i \leq f)$ is t_0.

Construct a schedule π' below: the jobs in $\mathcal{F}_i$ $(1 \leq i \leq f)$ form a batch starting at time ${s_i}'$, where:

$${s_i}' = \begin{cases} (1+\alpha)^{i-(k+1)}t_0, & k+1 \leq i \leq f \\ \max\{s_i+\epsilon,(1+\alpha)^{(f+i)-(k+1)}t_0\}, & 1 \leq i \leq k. \end{cases}$$

We can see that π' is feasible, and the maximum completion time of the jobs in π' is the completion time of the jobs in $\mathcal{F}_k$. So,

$$C_{\max}(\pi') = (1+\alpha)\max\{s_k+\epsilon,(1+\alpha)^{f-1}t_0\}. \tag{3}$$

Since $s_k = s$, we have $C_{\max}(\pi') = \max\{(1+\alpha)(s+\epsilon),(1+\alpha)^f t_0\}$. and $C_{\max}(\pi) \leq C_{\max}(\pi')$.
If $C_{\max}(\pi') = (1+\alpha)(s+\epsilon)$, then by Equation (2) we know that:

$$\frac{C_{\max}(\sigma)}{C_{\max}(\pi)} \geq \frac{(1+\alpha)^{f+1}s}{(1+\alpha)(s+\epsilon)} \to (1+\alpha)^f = (1+\alpha_{\max})^f,\ \epsilon \to 0.$$

If $C_{\max}(\pi') = (1+\alpha)^f t_0$, then by Equations (1) and (2), we know that:

$$\frac{C_{\max}(\sigma)}{C_{\max}(\pi)} \geq \frac{(1+\alpha)^{f+1}s}{(1+\alpha)^f t_0} = \frac{(1+\alpha)s}{t_0} \geq (1+\alpha)^f = (1+\alpha_{\max})^f.$$

Case 2 $s \geq (1+\alpha)^f t_0$.

According to the constructing of I, s_k is the last starting time of the jobs in time interval $[t_0,(1+\alpha)^f t_0)$, and $s_k+\epsilon$ is the last arrival time of all jobs.

Since s is the starting time of the first job whose completion time is at least $(1+\alpha)^f t_0$, we obtain $(1+\alpha)s_k < (1+\alpha)^f t_0$, i.e., $s_k < (1+\alpha)^{f-1} t_0$. From Equation (3), we have:

$$\begin{aligned} C_{\max}(\pi') &= \max\{(1+\alpha)(s_k+\epsilon), (1+\alpha)^f t_0\} \\ &\to \max\{(1+\alpha)s_k, (1+\alpha)^f t_0\} \\ &= (1+\alpha)^f t_0, \end{aligned}$$

as $\epsilon \to 0$.

By the definition of s, f jobs from f distinct job families have not been processed at time s. So,

$$C_{\max}(\sigma) \geq (1+\alpha)^f s.$$

Thus,

$$\frac{C_{\max}(\sigma)}{C_{\max}(\pi)} \geq \frac{(1+\alpha)^f s}{(1+\alpha)^f t_0} = \frac{s}{t_0} \geq (1+\alpha)^f = (1+\alpha_{\max})^f.$$

The result follows. □

Before introducing the online algorithm, given a batch B, we define some notations in the following:

$J(B)$: the last job with maximum deterioration rate in B.

$r(B)$: the arrival time of $J(B)$.

$s(B)$: the starting time of $J(B)$ in σ.

$\alpha(B)$: the deterioration rate of $J(B)$.

$U(t)$: the set of the unprocessed jobs at time t.

$B_i(t)$: the set of the unprocessed jobs of the same family at time t, $1 \leq i \leq f$, which is a waiting batch at time t if $B_i(t) \neq \emptyset$.

$\mathcal{B}(t)$: the set of the waiting batches at time t.

$|\mathcal{B}(t)|$: the number of all waiting batches at time t.

$r(\mathcal{B}(t)) = \min\{r(B) : B \in \mathcal{B}(t)\}$.

The online algorithm, called A_1 (Algorithm 1), can be stated as follows. Without causing confusion, assume that $B_i(t) = B_i$ in the following.

Algorithm 1: A_1

Input: Job instance I. **do**

Step 0: Set $t = t_0$.

Step 1: If $\mathcal{B}(t) = \emptyset$, then go to Step 5.

Step 2: Let $\mathcal{B}(t) = \{B_1, B_2, \cdots, B_k\}$ such that $\alpha(B_1) \geq \alpha(B_2) \geq \cdots \geq \alpha(B_k)$, where $k \leq f$.

Step 3: If $t \geq (1+\alpha(B_1))^k t_0$, then process the batch B_1 at time t. Reset $t = (1+\alpha(B_1))t$. Return to Step 1.

Step 4: If $t < (1+\alpha(B_1))^k t_0$, then reset $t = \min\{(1+\alpha(B_1))^k t_0, t^*\}$, where t^* is the arrival time of the next job. Go to Step 2.

Step 5: If new jobs arrive after t, then reset t as the arrival time of the first new job. Go to Step 1.

Output: Job schedule σ.

Example 1. *To make the algorithm more intuitive, we present an instance $\mathcal{I}_1$ in Table 1, where $\mathcal{F}_1$ and $\mathcal{F}_2$ are two different families. As shown in Figure 1, σ is the schedule generated by A_1 and π is an optimal offline schedule for $\mathcal{I}_1$, where $B_1 = \{J_1, J_3\}$ and $B_2 = \{J_2\}$.*

We have $C_{\max}(\sigma) = 81$ and $C_{\max}(\pi) = 9$.

Table 1. Instance $\mathcal{I}_1$.

Job	Arrival Time	Deterioration Rate
$J_1 \in \mathcal{F}_1$	$r_1 = t_0 = 1$	2
$J_2 \in \mathcal{F}_2$	$r_2 = t_0 = 1$	2
$J_3 \in \mathcal{F}_1$	$r_3 = 2$	1

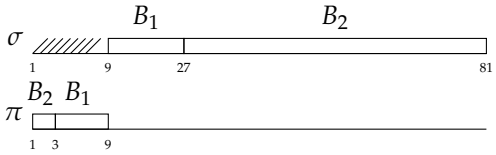

Figure 1. Schedule for Instance $\mathcal{I}_1$.

Suppose that r_l is the last arrival time. Let $s \geq r_l$ be the minimum time, such that s is the starting time of some batch and there is no idle time between s and $C_{\max}(\sigma)$ in σ. Let $\mathcal{B}$ be the set of the batches that process between s and $C_{\max}(\sigma)$ in σ and $s(\mathcal{B})$ be the start time of $\mathcal{B}$. Since $s(\mathcal{B}) = s \geq r_l$, each batch in $\mathcal{B}$ is from a different family and $\mathcal{B} = \mathcal{B}(s)$. From Algorithm 1, $\mathcal{B} = \{B_1, B_2, \cdots, B_k\}$ with $\alpha(B_1) \geq \alpha(B_2) \geq \cdots \geq \alpha(B_k)$, where $k \leq f$. Then,

$$C_{\max}(\sigma) = \prod_{i=1}^{k}(1 + \alpha(B_i))s(\mathcal{B}). \tag{4}$$

The following two lemmas are the competition ratio analyses of Algorithm 1.

Lemma 1. *Suppose the machine has an idle time immediately before $s(\mathcal{B})$ in σ. Then, $C_{\max}(\sigma)/C_{\max}(\pi) \leq (1 + \alpha_{\max})^f$.*

Proof. Since an idle time occurs immediately before $s(\mathcal{B})$ in σ, from Algorithm 1, we have: $s(\mathcal{B}) = \max\{r_l, (1 + \alpha(B_1))^k t_0\}$.

If $s(\mathcal{B}) = r_l$, then for each $B_i \in \mathcal{B}$ with $1 \leq i \leq k$, $J(B_i)$ arrives at time r_l. From Equation (4), we know that $C_{\max}(\sigma) = \prod_{i=1}^{k}(1 + \alpha(B_i))r_l \leq C_{\max}(\pi)$.

If $s(\mathcal{B}) = (1 + \alpha(B_1))^k t_0$, then from Equation (4) we have $C_{\max}(\sigma) = \prod_{i=1}^{k}(1 + \alpha(B_i))(1 + \alpha(B_1))^k t_0$. Since $C_{\max}(\pi) \geq \prod_{i=1}^{k}(1 + \alpha(B_i))t_0$, so $C_{\max}(\sigma)/C_{\max}(\pi) \leq (1 + \alpha(B_1))^k \leq (1 + \alpha_{\max})^f$. □

Lemma 2. *Suppose the machine has no idle time immediately before $s(\mathcal{B})$ in σ. Then, $C_{\max}(\sigma)/C_{\max}(\pi) \leq (1 + \alpha_{\max})^f$.*

Proof. Since the machine has no idle time immediately before $s(\mathcal{B})$ in σ, $s(\mathcal{B})$ is the completion time of some batch, say B^*, in σ. We have $s(\mathcal{B}) = (1 + \alpha(B^*))s(B^*)$. From the definition of s, we know $s(B^*) < r_l$.

We suppose that $\mathcal{B}$ is divided into two sets, $\mathcal{B}_1$ and $\mathcal{B}_2$, such that:

$$\mathcal{B}_1 = \{B_i \in \mathcal{B} : r(B_i) > s(B^*)\},$$
$$\mathcal{B}_2 = \{B_j \in \mathcal{B} : r(B_j) \leq s(B^*)\}.$$

Since $\mathcal{B} = \mathcal{B}_1 \bigcup \mathcal{B}_2$, $s(\mathcal{B}) = (1 + \alpha(B^*))s(B^*)$, then from Equation (4) we have:

$$\begin{aligned} C_{\max}(\sigma) &= \prod_{i=1}^{k}(1+\alpha(B_i))s(\mathcal{B}) \\ &= \prod_{B_i\in\mathcal{B}}(1+\alpha(B_i))s(\mathcal{B}) \\ &= \prod_{B_i\in\mathcal{B}_1}(1+\alpha(B_i))\prod_{B_j\in\mathcal{B}_2}(1+\alpha(B_j))(1+\alpha(B^*))s(B^*). \end{aligned}$$

From the definition of $\mathcal{B}_1$, we know $C_{\max}(\pi) \geq \prod_{B_i\in\mathcal{B}_1}(1+\alpha(B_i))s(B^*)$. Hence,

$$\begin{aligned} C_{\max}(\sigma) &= \prod_{B_i\in\mathcal{B}_1}(1+\alpha(B_i))\prod_{B_j\in\mathcal{B}_2}(1+\alpha(B_j))(1+\alpha(B^*))s(B^*) \\ &\leq \prod_{B_j\in\mathcal{B}_2}(1+\alpha(B_j))(1+\alpha(B^*))C_{\max}(\pi). \end{aligned}$$

According to the definition of $\mathcal{B}_2$ and Algorithm 1, each batch in set $\mathcal{B}_2$ and B^* belongs to the different family. Then, at most $f-1$ batches exist in $\mathcal{B}_2$. Hence,

$$C_{\max}(\sigma)/C_{\max}(\pi) \leq \prod_{B_j\in\mathcal{B}_2}(1+\alpha(B_j))(1+\alpha(B^*)) \leq (1+\alpha_{\max})^f.$$

□

By Lemmas 1 and 2, and Theorem 1, we can reach the final conclusion.

Theorem 2. *For problem* $1|online,r_j,p\text{-}batch,b=\infty,f\text{-}family,p_j=\alpha_j t|C_{\max}$, *Algorithm 1 has a competitive ratio of* $(1+\alpha_{\max})^f$ *and is the best possible.*

3. Parallel Batch Machines ($m \geq 1$)

In this section, we consider the online scheduling on m parallel batch machines and the jobs belong to m incompatible deteriorating job families. We prove the lower bound and construct the best possible online algorithm with a competitive ratio of $1+\alpha_{\max}$.

Theorem 3. *For problem* $Pm|online,r_j,p\text{-}batch,b=\infty,m\text{-}family,p_j=\alpha_j t|C_{\max}$, *no online algorithm exists with a competitive ratio less than* $1+\alpha_{\max}$.

Proof. Let H be any online algorithm and I be a job instance provided by the adversary. In the instance I, all the jobs have a deterioration rate of α.

At time t_0, m jobs $J_1, J_2, \cdots, J_m$ from different families arrive. Suppose that job J_j starts processing at time s_j in σ, $j=1,2,\cdots,m$.

If a job J_k exists such that $s_k \geq (1+\alpha)t_0$, where $1 \leq k \leq m$, then the adversary does not release other jobs. Hence,

$$C_{\max}(\sigma) \geq (1+\alpha)s_k \geq (1+\alpha)^2 t_0 \text{ and } C_{\max}(\pi) = (1+\alpha)t_0.$$

We have $C_{\max}(\sigma)/C_{\max}(\pi) \geq 1+\alpha = 1+\alpha_{\max}$.

If for each job J_j with $1 \leq j \leq m$, $s_j < (1+\alpha)t_0$, let $J_l \in \{J_1, J_2, \cdots, J_m\}$ is the last starting job. At time $s_l+\epsilon$, a copy of the job $J_j(\ j=1,2,\cdots,m)$ arrives. We have:

$$C_{\max}(\sigma) \geq (1+\alpha)^2 s_l \text{ and } C_{\max}(\pi) = (1+\alpha)(s_l+\epsilon).$$

Hence, $C_{\max}(\sigma)/C_{\max}(\pi) \geq (1+\alpha)s_l/(s_l+\epsilon) \to 1+\alpha = 1+\alpha_{\max}$, as $\epsilon \to 0$.

The result follows. □

Before providing the online algorithm, we define some notations used in the following:

$U(t)$: the set of the unprocessed jobs at time t.

$\alpha_{\max}(t)$: the maximum deterioration rate of the jobs arrived at t or before t.

$m(t)$: the number of the idle machines at time t.

$f(t)$: the number of job families in $U(t)$ at time t.

$B_i(t)$: the nonempty set of the unprocessed jobs of the same family at time t, where $1 \leq i \leq f(t)$.

$J^i(t)$: the job with the maximum deterioration rate in $B_i(t)$, where $1 \leq i \leq f(t)$.

$\alpha^i(t)$: the deterioration rate of the job $J^i(t)$, where $1 \leq i \leq f(t)$.

Without loss of generality, assume that $\alpha^1(t) \geq \alpha^2(t) \geq \cdots \geq \alpha^{f(t)}(t)$. The online algorithm, called A_2 (Algorithm 2), can be stated as follows.

Algorithm 2: A_2

Input: Job instance I. **do**

Step 0: Set $t = t_0$.

Step 1: If $U(t) = \emptyset$, then go to Step 6.

Step 2: If $m(t) = m$, and $t \geq (1 + \alpha_{\max}(t))t_0$, then at time t, start $B_i(t)$ as a single batch on the idle machine for any $i = 1, 2, \cdots, f(t)$. Reset $t = (1 + \alpha^{f(t)}(t))t$. Go to Step 1.

Step 3: If $m(t) = m$, and $t < (1 + \alpha_{\max}(t))t_0$, then reset $t = t^*$, such that t^* is either the arrival time of the next job or $(1 + \alpha_{\max}(t))t_0$. Go to Step 1.

Step 4: If $m(t) < m$, and $t \geq (1 + \alpha_{\max}(t))(1 + \alpha^1(t))t_0$, then at time t, start $B_i(t)$ as a single batch on the idle machine for any $i = 1, 2, \cdots, \min\{m(t), f(t)\}$. Reset $t = t^*$, such that t^* is either the arrival time of the next job or $(1 + \alpha^{\min\{m(t), f(t)\}}(t))t$. Go to Step 1.

STEP 5: If $m(t) < m$, and $t < (1 + \alpha_{\max}(t))(1 + \alpha^1(t))t_0$, then reset $t = t^*$, such that either t^* is the arrival time of the next job or $m(t^*) > m(t)$, or $t^* = (1 + \alpha_{\max}(t))(1 + \alpha^1(t))t_0$.

Step 6: If new jobs arrive after t, then reset t as the arrival time of the first new job. Go to Step 1.

Output: Job schedule σ.

Example 2. *To make the algorithm more intuitive, we present an instance $\mathcal{I}_2$ in Table 2. Figure 2 depicts the schedule generated by Algorithm 2 and Figure 3 is an optimal offline schedule for $\mathcal{I}_2$.*

We have $C_{\max}(\sigma) = 12$ and $C_{\max}(\pi) = 8$.

Table 2. Instance $\mathcal{I}_2$.

Job	Arrival Time	Deterioration Rate
$J_1 \in \mathcal{F}_1$	$r_1 = t_0 = 1$	2
$J_2 \in \mathcal{F}_2$	$r_2 = t_0 = 1$	1
$J_3 \in \mathcal{F}_3$	$r_3 = t_0 = 1$	1
$J_4 \in \mathcal{F}_1$	$r_4 = 4$	1

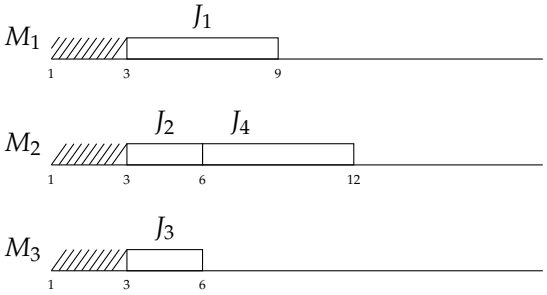

Figure 2. Schedule generated by A_2 for Instance $\mathcal{I}_2$.

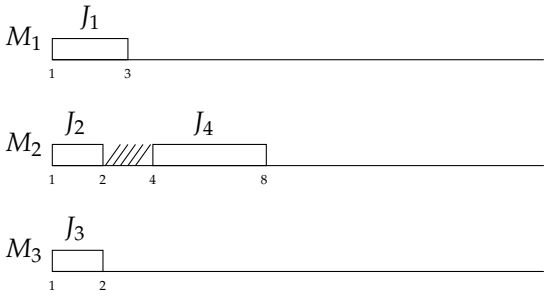

Figure 3. Optimal offline schedule for Instance $\mathcal{I}_2$.

Suppose that Algorithm 2 generates n batches $B_1, B_2, \cdots B_n$. For batch B_i, we define some notations in the following:

J_i: the job with the maximum deterioration rate in B_i.

α_i: the deterioration rate of J_i or the deterioration rate of B_i.

r_i: the arrival time of J_i.

s_i: the starting time of B_i in σ, suppose that $s_1 \leq s_2 \leq \cdots \leq s_n$.

The following is the competition ratio analysis of the Algorithm 2.

Let B_l be the first batch in σ assuming the objective value. B_i is a regular batch if $s_i = \max\{(1+\alpha_{\max}(s_i))t_0, r_i\}$ or $\max\{(1+\alpha_{\max}(s_i))t_0, r_i\} < s_i \leq (1+\alpha_{\max}(s_i))(1+\alpha^1(s_i))t_0$.

Lemma 3. *Suppose that only one job J_i exists in batch B_i of σ, $i = 1, 2, \cdots, n$, then the value of $C_{\max}(\sigma)/C_{\max}(\pi)$ does not decrease.*

Proof. From Algorithm 2, the start time of batch B_i is only related to the maximum deterioration rate of the jobs in this batch and the maximum deterioration rate of all jobs that have arrived. So, the value of $C_{\max}(\sigma)$ does not change when we assume each batch B_i has only one job J_i. The reduction in the number of jobs may decrease the value of $C_{\max}(\pi)$, so the value of $C_{\max}(\sigma)/C_{\max}(\pi)$ does not decrease. □

In the following, we assume that only one job J_i exists in batch B_i of σ, $i = 1, 2, \cdots, n$. Per Lemma 3, this does not influence the competition ratio analysis of Algorithm 2.

Lemma 4. $\alpha_l = \alpha^1(s_l)$.

Proof. Obviously, $\alpha_l \leq \alpha^1(s_l)$. If $\alpha_l < \alpha^1(s_l)$, since $J^1(s_l) \in U(s_l)$, then

$$C_{\max}(\sigma) \geq (1+\alpha^1(s_l))s_l > (1+\alpha_l)s_l.$$

This contradicts the completion time of B_l being the maximum completion time. Hence, $\alpha_l = \alpha^1(s_l)$. □

Lemma 5. *If the batch B_l is a regular batch, then $C_{\max}(\sigma)/C_{\max}(\pi) \leq 1+\alpha_{\max}$.*

Proof. Since B_l is a regular batch, then

$$s_l = \max\{(1+\alpha_{\max}(s_l))t_0, r_l\},$$

or

$$\max\{(1+\alpha_{\max}(s_l))t_0, r_l\} < s_l \leq (1+\alpha_{\max}(s_l))(1+\alpha^1(s_l))t_0.$$

Case 1 $s_l = \max\{(1+\alpha_{\max}(s_l))t_0, r_l\}$.

Then

$$\begin{aligned} C_{\max}(\sigma) &= (1+\alpha_l)s_l \\ &= \max\{(1+\alpha_l)(1+\alpha_{\max}(s_l))t_0, (1+\alpha_l)r_l\} \\ &\leq (1+\alpha_{\max})\max\{(1+\alpha_l)t_0, r_l\} \\ &\leq (1+\alpha_{\max})C_{\max}(\pi). \end{aligned}$$

Case 2 $\max\{(1+\alpha_{\max}(s_l))t_0, r_l\} < s_l \leq (1+\alpha_{\max}(s_l))(1+\alpha^1(s_l))t_0$.

In this case, at time s_l, some batches must have a start time less than s_l being processed. Let B_a be the last such batch to start, then $s_a < s_l \leq (1+\alpha_a)s_a$. Hence,

$$C_{\max}(\sigma) = (1+\alpha_l)s_l \leq (1+\alpha_l)(1+\alpha_a)s_a. \tag{5}$$

Suppose that $r_l \leq s_a$. Since $s_l > s_a$, the batch with a larger deterioration rate has higher priority in σ, then $\alpha_l \leq \alpha_a \leq \alpha^1(s_a)$ per Algorithm 2. At time s_a, J_l does not start processing. This indicates that there is no machine that can process J_l at time s_a, i.e., $m(s_a) < f(s_a) \leq m$. From Algorithm 2, we know that $s_a \geq (1+\alpha_{\max}(s_a))(1+\alpha^1(s_a))t_0$. By Lemma 4, we have $\alpha_l = \alpha^1(s_l)$. Hence,

$$\alpha^1(s_a) \geq \alpha_a \geq \alpha_l = \alpha^1(s_l).$$

By the definition of B_a, we have $\alpha_{\max}(s_a) = \alpha_{\max}(s_l)$, so

$$s_a \geq (1+\alpha_{\max}(s_a))(1+\alpha^1(s_a))t_0 \geq (1+\alpha_{\max}(s_l))(1+\alpha^1(s_l))t_0 \geq s_l.$$

This contradicts $s_a < s_l$. Hence $r_l > s_a$.
Thus, $C_{\max}(\pi) \geq (1+\alpha_l)r_l > (1+\alpha_l)s_a$. From Equation (5), we have

$$C_{\max}(\sigma)/C_{\max}(\pi) < 1+\alpha_a \leq 1+\alpha_{\max}.$$

□

In the following, we discuss the case where B_l is not a regular batch. This implies that no machine is idle immediately before time s_l, where $s_l > \max\{(1+\alpha_{\max}(s_l))(1+\alpha^1(s_l)t_0, r_l\}$. Renumber the m last batches starting on the m machines before time s_l to $B_{l,1}, B_{l,2}, \cdots, B_{l,m}$, such that $s_{l,1} \leq s_{l,2} \leq \cdots \leq s_{l,m}$. By Lemma 4, we have $\alpha_l = \alpha^1(s_l)$. So,

$$C_{\max}(\sigma) = (1+\alpha_l)s_l \leq (1+\alpha_l)\min_{1\leq j\leq m}\{(1+\alpha_{l,j})s_{l,j}\}. \tag{6}$$

If $s_{l,1} = s_{l,2} = \cdots = s_{l,k} = \cdots = s_{l,m} < s_l$, then $J_{l,1}, J_{l,2}, \cdots, J_{l,m}$ belong to m different job families and one of them belongs to the same family with J_l. Then $r_l > s_{l,1}$ and $C_{\max}(\pi) \geq (1+\alpha_l)r_l > (1+\alpha_l)s_{l,1}$. From Equation (6), we have:

$$C_{\max}(\sigma) \leq (1+\alpha_l)\min_{1\leq j\leq m}\{(1+\alpha_{l,j})s_{l,1}\} < \min_{1\leq j\leq m}(1+\alpha_{l,j}) \leq 1+\alpha_m.$$

In the following, we suppose that $s_{l,i} < s_{l,i+1}$ for some $i \in \{1, 2, \cdots, m-1\}$. Let k be the index that satisfies $s_{l,1} = s_{l,2} = \cdots = s_{l,k} < s_{l,k+1} \leq \cdots \leq s_{l,m} < s_l$, then $\alpha_{l,1} \geq \alpha_{l,2} \geq \cdots \geq \alpha_{l,k}$ and $J_{l,1} = J^1(s_{l,1})$. If $k \geq 2$, then we observe that any two jobs from $\{J_{l,1}, J_{l,2}, \cdots, J_{l,k}\}$ belong to different job families. Define

$$I_1 = \{J_{l,1}, J_{l,2}, \cdots, J_{l,k}\} \text{ and } I_2 = \{J_{l,k+1}, \cdots, J_{l,m}\}.$$

Lemma 6. *For any job $J_{l,j} \in I_2$, we have $s_{l,j} \geq (1+\alpha_{\max}(s_{l,j}))(1+\alpha_{l,j})t_0$.*

Proof. Since $s_{l,1} = s_{l,2} = \cdots = s_{l,k} < s_{l,k+1} \leq \cdots \leq s_{l,m} < s_l$, there is no idle machine immediately before time s_l, and $B_{l,1}, B_{l,2}, \cdots, B_{l,m}$ is the m last batches starting on the m machines before time s_l, then $m(t) < m$ for any time $t \in [s_{l,k+1}, s_{l,m}]$. From Algorithm 2, we have: $s_{l,j} \geq (1+\alpha_{\max}(s_{l,j}))(1+\alpha_{l,j})t_0$ for any job $J_{l,j} \in I_2$. □

Lemma 7. *If* $r_l > s_{l,k+1}$, *then* $C_{\max}(\sigma)/C_{\max}(\pi) \leq 1+\alpha_{\max}$.

Proof. Since $r_l > s_{l,k+1}$, then $C_{\max}(\pi) \geq (1+\alpha_l)r_l > (1+\alpha_l)s_{l,k+1}$. From Equation (6), we have:

$$C_{\max}(\sigma) \leq (1+\alpha_l) \min_{1\leq j\leq m}\{(1+\alpha_{l,j})s_{l,j}\} \leq (1+\alpha_l)(1+\alpha_{l,k+1})s_{l,k+1}.$$

Hence,

$$C_{\max}(\sigma)/C_{\max}(\pi) < 1+\alpha_{l,k+1} \leq 1+\alpha_{\max}.$$

□

Lemma 8. *If* $r_l \leq s_{l,k+1}$, *then* $C_{\max}(\sigma)/C_{\max}(\pi) \leq 1+\alpha_{\max}$.

Proof. Since $r_l \leq s_{l,k+1}$, from Algorithm 2, we have:

$$\alpha_l \leq \min\{\alpha_{l,j} | J_{l,j} \in I_2\}. \tag{7}$$

If a job $J_{l,h} \in I_2 \backslash \{J_{l,k+1}\}$ exists such that $r_{l,h} > s_{l,k+1}$, then from Equation (7), we obtain:

$$C_{\max}(\pi) \geq (1+\alpha_{l,h})r_{l,h} > (1+\alpha_{l,h})s_{l,k+1} \geq (1+\alpha_l)s_{l,k+1}.$$

From Equation (6), we have:

$$C_{\max}(\sigma) \leq (1+\alpha_l) \min_{1\leq j\leq m}\{(1+\alpha_{l,j})s_{l,j}\} \leq (1+\alpha_l)(1+\alpha_{l,k+1})s_{l,k+1}.$$

Hence,

$$C_{\max}(\sigma)/C_{\max}(\pi) < 1+\alpha_{l,k+1} \leq 1+\alpha_{\max}.$$

Suppose that, for any job $J_{l,h} \in I_2 \backslash \{J_{l,k+1}\}$, $r_{l,h} \leq s_{l,k+1}$. Since $r_{l,k+1} \leq s_{l,k+1}$ and $r_l \leq s_{l,k+1}$, then the arrival time of all jobs from $I_2 \bigcup \{J_l\}$ is less than $s_{l,k+1}$. Thus, any two jobs from $I_2 \bigcup \{J_l\}$ belong to distinct job families.

Claim At least one job in $I_2 \bigcup \{J_l\}$ has an arrival time greater than $s_{l,k}$.

Otherwise, if the arrival time of all jobs is less than or equal to $s_{l,k}$, then all jobs in $I_1 \bigcup I_2 \bigcup \{J_l\}$ are available at time $s_{l,1}$, and each job independently forms a batch in σ. We obtain that every two jobs from $I_1 \bigcup I_2 \bigcup \{J_l\}$ belong to distinct job families. Since $I_1 \bigcup I_2 \bigcup \{J_l\} = \{J_{l,1}, J_{l,2}, \cdots, J_{l,m}\} \bigcup \{J_l\}$, then $f(s_{l,1}) = m+1 > m$. This contradicts $f(s_{l,1}) \leq m$. The claim follows.

Since at least one job from $I_2 \bigcup \{J_l\}$ arrives after $s_{l,k}$, from Equation (7), we have:

$$C_{\max}(\pi) > (1+\alpha_l)s_{l,k}.$$

From Equation (6), we have:

$$C_{\max}(\sigma) \leq (1+\alpha_l) \min_{1\leq j\leq m}\{(1+\alpha_{l,j})s_{l,j}\} \leq (1+\alpha_l)(1+\alpha_{l,k})s_{l,k}.$$

Hence,

$$C_{\max}(\sigma)/C_{\max}(\pi) < 1+\alpha_{l,k} \leq 1+\alpha_{\max}.$$

□

From Lemmas 5, 7 and 8, and Theorem 3, we obtain the following theorem.

Theorem 4. *For problem* $Pm|online,r_j,p\text{-}batch,b = \infty,m\text{-}family,p_j = \alpha_j t|C_{\max}$, *Algorithm 2 has a competitive ratio of* $1+\alpha_{\max}$ *and is the best possible.*

4. Conclusions and Future Research

In this paper, we outlined two best possible online algorithms. The first algorithm for problem $1|online,r_j,p\text{-}batch,b = \infty,f\text{-}family,p_j = \alpha_j t|C_{\max}$ is a simple delay algorithm. We obtained the delay time by analyzing the properties of the unprocessed jobs, providing the best possible online algorithm with the competitive ratio of $(1+\alpha_{\max})^f$. The second algorithm for problem $Pm|online,r_j,p\text{-}batch,b = \infty,m\text{-}family,p_j = \alpha_j t|C_{\max}$ is a more complex delay algorithm. We obtained the different delay times depending on the number of idle machines and provide the best possible online algorithm with the competitive ratio of $1+\alpha_{\max}$. The results are shown in Table 3.

Table 3. Summary of results.

Parallel Machine	Number of Families	Optimum Rate
$m = 1$	f	$(1+\alpha_{\max})^f$; best possible
$m \geq 1$	$f = m$	$1+\alpha_{\max}$; best possible

In future research, the general linear deterioration effect, such as $p_j = \alpha_j s_j + \beta_j$, is worthy of research. In additional, for the online scheduling problem on m parallel machines with linear deteriorating jobs to minimize the makespan, Yu et al. [22] only proved that no deterministic online algorithm is better than $(1+\alpha_{\max})$-competitive when $m = 2$, where $\alpha_{\max}$ is the maximum deterioration rate of all jobs. However, no best possible online algorithm has been reported. This is also a topic for further study.

Author Contributions: Conceptualization, methodology and funding acquisition, W.L.; formal analysis and writing-original draft preparation, L.W.; writing—review, supervision and project administration, W.L. and X.C.; investigation, H.Y. All authors have read and agreed to the published version of the manuscript.

Funding: Research supported by NSFC (Nos. 11571321,11971443 and 11771406).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Borodin, A.; El-Yaniv, R. *Online Computation and Competitive Analysis*; Cambridge University Press: Cambridge, UK, 1998.
2. Brucker, P.; Gladky, A.; Hoogeveen, H.; Kovalyov, M.Y.; Potts, C.N.; Tautenhahn, T.; van de Velde, S.L. Scheduling a batching machine. *J. Sched.* **1998**, *1*, 31–54. [CrossRef]
3. Fu, R.Y.; Cheng, T.C.E.; Ng, C.T.; Yuan, J.J. An optimal online algorithm for single parallel-batch machine scheduling with incompatible job families to minimize makespan. *Oper. Res. Lett.* **2013**, *41*, 216–219. [CrossRef]
4. Li, W.H.; Yuan, J.J.; Yang, S.F. Online scheduling of incompatible unit-length job families with lookahead. *Theor. Comput. Sci.* **2014**, *543*, 120–125. [CrossRef]
5. Tian, J.; Cheng, T.C.E.; Ng, C.T.; Yuan, J.J. Online scheduling on unbounded parallel-batch machines with incompatible job families. *Theor. Comput. Sci.* **2011**, *412*, 2380–2386. [CrossRef]
6. Kunnathur, A.S.; Gupta, S.K. Minimizing the makespan with late start penalties added to processing times in a single facility scheduling problem. *Eur. J. Oper. Res.* **1990**, *47*, 56–64. [CrossRef]
7. Mosheiov, G. Schedulig jobs under simple linear deterioration. *Comput. Oper. Res.* **1994**, *21*, 653–659. [CrossRef]

8. Browne, S.; Yechiali, U. Scheduling deteriorating jobs on a single processor. *Oper. Res.* **1990**, *38*, 495–498. [CrossRef]
9. Gupta, J.N.D.; Gupta, S.K. Single facility scheduling with nonlinear processing times. *Comput. Ind. Eng.* **1988**, *14*, 387–393. [CrossRef]
10. Gawiejnowicz, S.; Kononov, A. Isomorphic scheduling problems. *Ann. Oper. Res.* **2014**, *213*, 131–145. [CrossRef]
11. Gawiejnowicz, S. Scheduling deteriorating jobs subject to job or machine availability constraints. *Eur. J. Oper. Res.* **2007**, *180*, 472–478. [CrossRef]
12. Ji, M.; Cheng, T.C.E. Batch scheduling of simple linear deteriorating jobs on a single machine to minimize makespan. *Eur. J. Oper. Res.* **2010**, *202*, 90–98. [CrossRef]
13. Lee, W.; Wu, C.; Chung, Y. Scheduling deteriorating jobs on a single machine with release times. *Comput. Ind. Eng.* **2008**, *54*, 441–452. [CrossRef]
14. Ng, C.T.; Li, S.S.; Cheng, T.C.E.; Yuan, J.J. Preemptive scheduling with simple linear deterioration on a single machine. *Theor. Comput. Sci.* **2010**, *411*, 3578–3586. [CrossRef]
15. Ji, M.; He, Y.; Cheng, T.C.E. Scheduling linear deteriorating jobs with an availability constraint on a single machine. *Theor. Comput. Sci.* **2006**, *362*, 115–126. [CrossRef]
16. Agnetis, A.; Billaut, J.; Gawiejnowicz, S.; Pacciarelli, D.; Soukhal, A. *Multiagent Scheduling*; Springer: Berlin/Heidelberg, Germany, 2014.
17. Gawiejnowicz, S. *Time-Dependent Scheduling*; Springer: Berlin/Heidelberg, Germany, 2008.
18. Strusevich, V.; Rustogi, K. *Scheduling with Time-Changing Effects and Rate-Modifying Activities*; Springer: Berlin/Heidelberg, Germany, 2017.
19. Cheng, M.B.; Sun, S.J. A heuristic MBLS algorithm for two semi-online parallel machine scheduling problems with deterioration jobs. *J. Shanghai Univ.* **2007**, *11*, 451–456. [CrossRef]
20. Liu, M.; Zheng, F.; Wang, S.; Huo, J. Optimal algorithms for online single machine scheduling with deteriorating jobs. *Theor. Comput. Sci.* **2012**, *445*, 75–81. [CrossRef]
21. Ma, R.; Tao, J.P.; Yuan, J.J. Online scheduling with linear deteriorating jobs to minimize the total weighted completion time. *Appl. Math. Comput.* **2016**, *273*, 570–583. [CrossRef]
22. Yu, S.; Ojiaku, J.T.; Wong, P.W.H.; Xu, Y.F. Online makespan scheduling of linear deteriorating jobs on parallel machines. In Proceedings of the International Conference on Theory and Applications of Models of Computation 2012, Beijing, China, 16–21 May 2012; pp. 260–272.
23. Yu, S.; Wong, P.W.H. Online scheduling of simple linear deteriorating jobs to minimize the total general completion time. *Theor. Comput. Sci.* **2013**, *487*, 95–102. [CrossRef]
24. Graham, R.L.; Lawler, E.L.; Lenstra, J.K.; Kan, A.H.G.R. Optimization and approximation in deterministic sequencing and scheduling: A survey. *Ann. Discret.* **1979**, *5*, 646–675.