J Comb Optim (2016) 31:1609–1622
DOI 10.1007/s10878-015-9844-3

Online scheduling of equal length jobs on unbounded parallel batch processing machines with limited restart

Hailing Liu · **Jinjiang Yuan** · **Wenjie Li**

Published online: 26 February 2015

Abstract We consider the online scheduling of equal length jobs on unbounded parallel batch processing machines to minimize makespan with limited restart. In the problem m identical unbounded parallel batch processing machines are available to process the equal length jobs arriving over time. The processing batches are allowed limited restart. Here, "restart" means that a running task may be interrupted, losing all the work done on it, and the jobs in the interrupted task are then released and become independently unscheduled jobs, called restarted jobs. "Limited restart" means that only a running batch that contains no restarted jobs can be restarted. For this problem, we present a best possible online algorithm.

Keywords Online scheduling · Limited restart · Unit length jobs · Restricted batch

1 Introduction

In this paper, we consider the online scheduling of equal length jobs on m identical unbounded parallel batch processing machines with limited restart. In the online ver-

H. Liu · J. Yuan (✉)
School of Mathematics and Statistics, Zhengzhou University, Zhengzhou 450001, Henan, People's Republic of China
e-mail: yuanjj@zzu.edu.cn

H. Liu
College of Science, Henan Institute of Engineering, Zhengzhou 451191, Henan, People's Republic of China

W. Li
School of Mathematical Sciences, Luoyang Normal University, Luoyang 471022, Henan, People's Republic of China

sion, each job becomes available at its arrival time, which is unknown in advance, and its characteristics become known upon its arrival. A *parallel batch processing machine* is modeled as a system that can process up to b jobs simultaneously as a batch, where b is the batch capacity. The *processing time of a batch* is equal to the longest processing time of the jobs in the batch. All jobs processed in a batch start at the same time and complete at the same time. Depending on the characteristic of batch capacity b, there are two different models. One is the unbounded model, in which the capacity of batches is sufficiently large, i.e., $b = \infty$. The other is the bounded model, in which the capacity of batches is finite, i.e., $b < \infty$. In this paper we study the unbounded model, i.e., $b = \infty$. In our research, we assume that all jobs have equal processing times. By scaling, we may assume that each job has a processing time 1. The objective of the problem considered in this paper is to assign jobs to batches and determine their processing sequence on each machine so as to minimize the makespan, namely, the maximum completion time of all jobs.

Restart (see Hoogeveen et al. 2000) means that a running task may be interrupted, losing all the work done on it. The jobs in the interrupted task, which are called restarted jobs, are then released and become independently unscheduled jobs which can be scheduled later from scratch. Allowing restarts may give us a better schedule because we have the chance to change our original mind and make a better decision. For some scheduling models, we can obtain more efficient online algorithms by using restarts. For example, for online minimization of the maximum delivery time on a single machine using restarts, Van der Akker et al. (2003) gave an algorithm with competitive ratio $3/2$; while without restarts, a best possible online algorithm with competitive ratio $(\sqrt{5}+1)/2$ was given in Hoogeveen and Vestjens (2000). We also can see more research on online algorithms using restarts in Epstein and Stee (2003), Van Stee and La Poutré (2005), and Yuan et al. (2011).

Limited restart, which is first introduced in Fu et al. (2008), means that a job can be restarted at most once. So in the online parallel batch scheduling with limited restart, once a running batch contains some restarted jobs, we cannot interrupt the processing of the batch again. The assumption of limited restart is motivated by considering restarts as scarce resources. In practice, too many restarts of a job may cause the waste of resources and increase the probability of a spoiled product.

The quality of an online algorithm is measured by the *competitive ratio*. An online algorithm is called *ρ-competitive* if for any input instance, it generates a schedule with an objective value no worse than ρ times the value of an optimal off-line schedule. The nearer the ratio is to 1, the better the algorithm is.

Parallel batch scheduling is motivated by burn in operations in semiconductor manufacture (see Uzsoy et al. 1992, 1994). Online scheduling on parallel batch processing machines was first studied by Zhang et al. (2001) and Deng et al. (2003). They studied online scheduling problem to minimize makespan on a single unbounded parallel batch processing machine and independently presented the same best possible online algorithm which is $(\sqrt{5}+1)/2$-competitive. For the corresponding problem with bounded batch capacity, Poon and Yu (2005) showed that any FBLPT-based algorithm has competitive ratio at most 2 and for batch capacity 2, they gave an $7/4$-competitive online algorithm. Zhang et al. (2003) studied the online scheduling of equal length jobs on m parallel batching machines. They first presented a $(1+\beta_m)$-competitive optimal online algorithm for the unbounded version, where β_m is the positive solution of the equation $(1+\beta_m)^{m+1} = \beta_m + 2$. They also proposed a $(\sqrt{5}+1)/2$-competitive optimal online algorithm for the bounded version. For the problem of minimizing makespan on two unbounded parallel batch processing machines, Nong et al. (2008) proposed an online algorithm which is $\sqrt{2}$-competitive. And later Tian et al. (2009b) showed that $\sqrt{2}$ is the lower bound for the problem and gave a new optimal online algorithm. For the corresponding problem on m unbounded parallel batch processing machines, Liu et al. (2012) and Tian et al. (2009a) independently presented two different but best possible online algorithms which are $(1 + \alpha_m)$-competitive, where α_m is the positive solution of the equation $a_m^2 + m\alpha_m - 1 = 0$.

For minimizing makespan on an unbounded parallel batch processing machine using restarts, Fu et al. (2007) showed that there exists no on-line algorithm with a competitive ratio less than $(5 - \sqrt{5})/2$ and a best possible online algorithm matching the lower bound was presented in Yuan et al. (2011). Fu et al. (2008) studied the corresponding problem with limited restart and proposed a best possible online algorithm with competitive ratio 3/2. For minimizing makespan on two unbounded parallel batch processing machines using limited restart, a best possible online algorithm with competitive ratio $(\sqrt{3} + 1)/2$ was proposed in Fu et al. (2010) under the second-restart assumption. For minimizing makespan on a bounded parallel batch processing machine using restarts, an online algorithm with competitive ratio 3/2 was given in Chen et al. (2009). Recently, Liu and Yuan (2014) presented best possible online algorithms for minimizing makespan of equal length jobs on a bounded parallel batch processing machine with limited restart or restarts. More results of online scheduling in parallel batch machines can be found in Tian et al. (2014).

This paper studies online scheduling of equal length jobs on m unbounded parallel batch machines to minimize makespan with limited restart. In the scheduling notation, the problem is denoted by $Pm|\text{online}, \ r_j, \ p_j = 1, \ \text{p-batch}, \ b = \infty, \ \text{L-restart}|C_{\max}$.

The research approaches in this paper can be stated as follows. For each α with $0 < \alpha < 1$, we define an online algorithm LAZY(α). Based on algorithm LAZY(α), we generate a job instance $I(\alpha)$. The candidate choices of α in our research are given by $\alpha(i, j)$, where i, j are positive integers with $j < i \leq l_m$ and $\alpha(i, j)$ is the positive solution of equation $(1+x)^i - (1+x)^j = 1$. Here, l_m is a positive integer associated with m and will be defined later in the paper. Then we define

$$\alpha_m = \min_{0<j<i\leq l_m} \left\{\alpha(i,j) : \text{LAZY}(\alpha(i,j)) \text{ is } (1+\alpha(i,j))\text{-competitive on instance } I(\alpha(i,j))\right\}.$$

It is proved that α_m is also the minimum value of α with $0 < \alpha < 1$ so that LAZY(α) is $(1 + \alpha)$-competitive on instance $I(\alpha)$. Then we show that any online algorithm for problem $Pm|\text{online}, \ r_j, \ p_j = 1, \ \text{p-batch}, \ b = \infty, \ \text{L-restart}|C_{\max}$ has a competitive ratio of at least $1 + \alpha_m$. Finally, we present an $(1 + \alpha_m)$-competitive algorithm ALG$_m$ based on algorithm LAZY(α_m). This implies that ALG$_m$ is the best possible.

In Sect. 2, for each α with $0 < \alpha < 1$, we first present the online algorithm LAZY(α) and, based on the algorithm, we generate a job instance $I(\alpha)$ and a schedule σ_α. Some related properties are presented. In Sect. 3, we present the lower bound $1 + \alpha_m$ for the problem studied in this paper. In Sect. 4, we present the $(1 + \alpha_m)$-competitive algorithm ALG_m.

2 The Algorithm LAZY(α)

Throughout this paper, a *restricted batch* refers to a batch which contains at least one restarted job, and a *free batch* refers to a batch which contains no restarted jobs. This implies that, in an online algorithm, a running restricted batch cannot be interrupted again. Note that a running free batch can be interrupted freely and the batch capacity is unbounded. Then we take the convention that, in an online algorithm, if we start a restricted batch at a time S, then we will interrupt all free batches running at time S and generate a restricted batch which consists of all interrupted jobs and all other unscheduled available jobs at time S.

The following notations are used in our discussion:

- $F(i, j) = i + (i + 1) + \cdots + j$ for positive integers i, j with $i \le j$.
- β_m is the positive solution of equation $(1 + x)^{F(1,m+1)} - (1 + x)^{m+1} = 1$.
- $l_m = \min\{i : i \ge F(1, m + 1),\ \beta_m(1 + \beta_m)^i \ge 1\}$.
- $\alpha(i, j)$ is the positive solution of equation $(1 + x)^i - (1 + x)^j = 1$ where i and j are both positive integers with $i > j$. Then $\beta_m = \alpha(F(1, m + 1), m + 1)$.

Some properties of the above notations can be observed as follows which will be used in further discussions.

- For every two positive integers i and j with $i > j$, $(1 + x)^i - (1 + x)^j - 1$ is monotonically increasing in $x \ge 0$.
- $\alpha(i, j)$ is the unique positive solution of equation $(1 + x)^i - (1 + x)^j = 1$ with $i > j$. For $0 < \alpha < \alpha(i, j)$, $(1 + \alpha)^i - (1 + \alpha)^j < 1$.
- $\beta_m = \alpha(F(1, m + 1), m + 1)$ is monotonically decreasing in m.

When an online algorithm starts to process a batch B at a time t, we use the terminology "a job J arrives at time t" to denote the fact that job J arrives at time $t + \varepsilon$ for some very small positive number ε. In the discussion, we use t to replace $t + \varepsilon$ without loss of validity.

Since a job can be restarted once, the first starting time of a job is defined to be the starting time of the earliest starting batch including the job. Since the jobs have the equal processing time 1, the optimal value of any job instance is given by $r_{\max} + 1$ where $r_{\max}$ is the latest arriving time of jobs in the instance. Then we concentrate our attention on the schedules generated by online algorithms. Since the batch capacity is unbounded, we may assume that at most one batch starts at a time instant.

Let σ be an online schedule for a job instance. Let $S_1 < S_2 < \cdots < S_n$ be the sequence of time instants so that for each i with $1 \le i \le n$, a batch B_i starts at time S_i in σ. By omitting the machine information, the online schedule σ can be denoted by the sequence $\sigma = ((S_1, B_1), \ldots, (S_n, B_n))$. When no ambiguity can occur, we write $\sigma = (S_1, \ldots, S_n)$ for simplicity. We take the convention that the information of all starting batches (including the interrupted batches) is implied in σ. Thus, an online schedule is in fact the record of the implementation of an online algorithm. In the following, we just use "schedule" to denote "online schedule".

For convenience, we sometimes require that a batch B_i cannot be interrupted, no matter it contains restarted jobs or not. In this case, B_i is called an *assigned restricted batch*. In an online algorithm, we cannot interrupt the processing of a restricted or assigned restricted batch.

In the following discussion, we assume that m is an arbitrary fixed positive integer. For a given online algorithm H, we use the following algorithm, called Generating $\mathcal{I}(H)$, to generate a job instance $\mathcal{I}(H)$ together with the schedule generated by algorithm H on $\mathcal{I}(H)$. We use $B_i(H)$ to denote the i-th starting batch, $S_i(H)$ the starting time of $B_i(H)$ and $r_i(H)$ the latest arriving time of jobs in $B_i(H)$. $B'_i(H)$ is used to denote the i-th starting restricted batch, $S'_i(H)$ the starting time of $B'_i(H)$ and $r'_i(H)$ the latest arriving time of jobs in $B'_i(H)$. Furthermore, we use $B_{i,k}(H)$ to denote the i-th starting batch after $S'_k(H)$, $S_{i,k}(H)$ the starting time of $B_{i,k}(H)$ and $r_{i,k}(H)$ the latest arriving time of jobs in $B_{i,k}(H)$.

Generating $\mathcal{I}(H)$: Given any online algorithm H, we release one job at time 0, and then greedily release a new job whenever H starts a batch (that is, ε time after H starts a batch for some very small positive ε). We continue doing this until n jobs have been released in total, where n is sufficiently large for our discussion. The resulting input is denoted by $\mathcal{I}(H)$, and the schedule is denoted by σ_H. □

Note that Generating $\mathcal{I}(H)$ generates a job instance $\mathcal{I}(H) = \{J_1(H), \ldots, J_n(H)\}$ together with a schedule $\sigma_H = (S_1(H), \ldots, S_n(H))$ for $\mathcal{I}(H)$. We further define $S_0(H) = 0$ and $B_0(H) = \emptyset$.

For an online algorithm H and a positive number α, we say that σ_H is an *α-nice schedule* if all batches $B_i(H)$ satisfy $S_i(H) \le (1+\alpha)S_{i-1}(H)+\alpha$, i.e., $S_i(H)+1 \le (1+\alpha)(S_{i-1}(H)+1)$. If $S_i(H) = (1+\alpha)S_{i-1}(H)+\alpha$, i.e., $S_i(H)+1 = (1+\alpha)(S_{i-1}(H)+1)$, then $B_i(H)$ is called an *α-regular batch* in σ_H. Furthermore, σ_H is called an *α-regular schedule* if all batches in σ_H are α-regular.

Let $U(t)$ be the set of the available unscheduled jobs at time t in an online algorithm. Write $r(t) = \max\{r_j : J_j \in U(t)\}$. Let α be a real number with $0 < \alpha < 1$. The online algorithm, called LAZY(α), can be described as follows.

Algorithm LAZY(α):

1. Wait until the time t so that $U(t) \ne \emptyset$.
2. Wait until the current time $t = (1+\alpha)r(t)+\alpha$.
3. If all machines are busy at time t, stop all free batches (if any).
4. Wait until either a machine is available or a new job arrives.
5. If no new jobs arrive during the waiting procedure in step 4, start to process all unfinished jobs in one batch on an available machine. Go to step 1.
6. If a new job arrives during the waiting procedure in step 4, go to step 2.

□

Note that, for the instance consisting of all jobs released by a time t, the optimal off-line schedule has makespan $r(t)+1$. Therefore, if LAZY(α) ever waits in step 4 and a batch is processed in step 5, then some batch starts at a time $t > (1+\alpha)r(t)+\alpha$, and so, LAZY$(\alpha)$ is not $(1+\alpha)$-competitive.

By running algorithm LAZY(α) for instance $\mathcal{I}$(LAZY(α)), we get a schedule $\sigma_{\text{LAZY}(\alpha)}$. By algorithm LAZY($\alpha$), we have $S_i(\text{LAZY}(\alpha)) \geq (1+\alpha)S_{i-1}(\text{LAZY}(\alpha))+\alpha$ for $i \geq 1$. Since $S_0(\text{LAZY}(\alpha)) = 0$, this further implies that $S_i(\text{LAZY}(\alpha)) \geq (1+\alpha)^i - 1$ for $i \geq 1$.

The following notations are also used in our discussion.

- $\mathcal{I}(\alpha)$ is the job instance consisting of the first l_m jobs of $\mathcal{I}$(LAZY(α)).
- σ_α is the subschedule of $\sigma_{\text{LAZY}(\alpha)}$ for instance $\mathcal{I}(\alpha)$. Then there are total l_m batches in σ_α.
- Based on $\sigma(\alpha)$, we define $B_i(\alpha)$, $S_i(\alpha)$, etc. completely analogously to $B_i(H)$, $S_i(H)$, etc..
- $S_0(\alpha) = S'_0(\alpha) = 0$.
- $S_{0,k}(\alpha) = S'_k(\alpha)$.

Note that $\sigma_\alpha = (S_1(\alpha), \ldots, S_{l_m}(\alpha))$. We first prove several useful inequalities.

Recall that, for every two positive integers i and j with $i > j$, $(1+x)^i - (1+x)^j - 1 = (1+x)^j((1+x)^{i-j} - 1) - 1$ is monotonically increasing in $x \geq 0$ and $\alpha(i, j)$ is the unique positive solution of equation $(1+x)^j((1+x)^{i-j} - 1) - 1 = 0$. So, if (i, j) and (i', j') are two pairs of positive integers with $i > j$ and $i' > j'$, then $(j, i-j) \leq (j', i'-j')$ implies that $\alpha(i, j) \geq \alpha(i', j')$. $(j, i-j) \leq (j', i'-j')$ and $(j, i-j) \neq (j', i'-j')$ imply that $\alpha(i, j) > \alpha(i', j')$. Applying this observation to the three pairs $(m+1, 1)$, $(2m+1, 2)$, and $(3m, m+1)$, we can deduce that

$$\alpha(m+1, 1) > \alpha(2m+1, 2) \geq \alpha(3m, m+1). \tag{1}$$

Similarly, applying the above observation to the two pairs $(2m+1, 2)$ and $(F(1, m+1), m+1)$ by noting that $2m-1 \leq F(1, m) = F(1, m+1) - (m+1)$, we can deduce that

$$\beta_m = \alpha(F(1, m+1), m+1) \leq \alpha(2m+1, 2), \tag{2}$$

where the equality in (2) follows from the definition of β_m.

Lemma 2.1 *Let α be a real number with $0 < \alpha < 1$. Then σ_α has the following properties.*

(i) *Every batch $B_k(\alpha)$ with $S_k(\alpha) \leq S'_m(\alpha)$ is α-regular in σ_α and $S_k(\alpha) = (1+\alpha)^k - 1$.*
(ii) *If $\alpha \leq \alpha(2m+1, 2)$, then $B'_1(\alpha)$ exists and $B'_1(\alpha) = B_{m+1}(\alpha)$.*
(iii) *If $\alpha \geq \alpha(2m+1, 2)$, then σ_α is an α-regular schedule.*

Proof By the definition of LAZY(α) and $I(\alpha)$, we have $S_0(\alpha) = 0$ and $S_k(\alpha) = (1+\alpha)S_{k-1}(\alpha) + \alpha$ as long as LAZY(α) does not wait in Step 4. But LAZY(α) only waits in Step 4 if all machines are running restricted batches at the time it enters Step 4, which can happen at the earliest after the mth restricted batch has started. Now induction on k proves (i).

To prove (ii), suppose that α is an arbitrary positive number with $\alpha \leq \alpha(2m+1, 2)$. From (1), we have $\alpha(2m+1, 2) < \alpha(m+1, 1)$, and so, $\alpha < \alpha(m+1, 1)$. Since $\alpha(m+1, 1)$ is the positive solution of equation $(1+x)^{m+1} - (1+x) = 1$ and $(1+$

$x)^{m+1} - (1+x)$ is monotonically increasing in $x > 0$, we have $(1+\alpha)^{m+1} - (1+\alpha) < 1$. From property (i), $B_i(\alpha)$ $(1 \le i \le m)$ are all α-regular with $S_i(\alpha) = (1+\alpha)^i - 1$. Thus $(1+\alpha)S_m(\alpha) + \alpha - (S_1(\alpha) + 1) = (1+\alpha)^{m+1} - (1+\alpha) - 1 < 0$. This implies that at time $(1+\alpha)S_m(\alpha) + \alpha$, $B_i(\alpha)$ $(1 \le i \le m)$ are m free batches running on different machines and all uncompleted. So by algorithm LAZY(α), $B_{m+1}(\alpha)$ is a restricted batch and $B'_1(\alpha) = B_{m+1}(\alpha)$.

To prove (iii), let α be a positive number with $\alpha \ge \alpha(2m+1, 2)$. From (1), we have $\alpha(2m+1, 2) \ge \alpha(3m, m+1)$, and so, $\alpha \ge \alpha(3m, m+1)$. If there are no restricted batches in σ_α, the result holds trivially. Thus we may assume that there are some restricted batches in σ_α. If possible let $k \le l_m$ be the minimum so that $B_k(\alpha)$ is not α-regular. Then $S_k(\alpha) > (1+\alpha)S_{k-1}(\alpha) + \alpha$ and $B_i(\alpha)$, $1 \le i \le k-1$, are α-regular. From the fact $S_0(\alpha) = 0$, we can inductively deduce that $S_i(\alpha) = (1+\alpha)^i - 1$ for $1 \le i \le k-1$.

The choice of k also implies that, at time $t = (1+\alpha)S_{k-1}(\alpha) + \alpha$, each machine is occupied by a restricted batch. Suppose that $B_{k_1}(\alpha), \ldots, B_{k_m}(\alpha)$ are the m restricted batches running at time t with $k_1 < \cdots < k_m = k-1$. Then $t < S_{k_1}(\alpha) + 1$. By the definition of $\mathcal{I}(\alpha)$, for each i with $2 \le i \le m$, $B_{k_i - 1}(\alpha)$ is a free batch interrupted by $B_{k_i}(\alpha)$ at time $S_{k_i}(\alpha)$. Then $k_i \ge k_{i-1} + 2$ for $2 \le i \le m$. It follows that $k_m \ge k_1 + 2(m-1)$.

Now $S_{k-1}(\alpha)+1 = S_{k_m}(\alpha)+1 = (1+\alpha)^{k_m - k_1}(S_{k_1}(\alpha)+1) \ge (1+\alpha)^{2m-2}(S_{k_1}(\alpha)+1)$. Then $t = (1+\alpha)(S_{k-1}(\alpha)+1) - 1 \ge (1+\alpha)^{2m-1}(S_{k_1}(\alpha)+1) - 1$. Since $t < S_{k_1}(\alpha) + 1$, then $((1+\alpha)^{2m-1} - 1)(S_{k_1}(\alpha)+1) < 1$. As $S_{k_1}(\alpha) \ge S'_1(\alpha) \ge S_{m+1}(\alpha) = (1+\alpha)^{m+1} - 1$, we have $(1+\alpha)^{3m} - (1+\alpha)^{m+1} < 1$. This implies that $\alpha < \alpha(3m, m+1)$, a contradiction. The proof is completed. □

Lemma 2.2 *Let α be a real number with $0 < \alpha < \beta_m$. Then, for every online algorithm A, σ_A is not an α-nice schedule. Moreover, σ_α is not an α-nice schedule.*

Proof From (2), we have $\beta_m \le \alpha(2m+1, 2)$. Let A be an arbitrary online algorithm. We consider the job instance $\mathcal{I}(A)$ and the schedule σ_A generated by algorithm Generating $\mathcal{I}(A)$ so that we have sufficiently many batches in σ_A.

Let σ_A^1 be the subschedule of σ_A consisting of the first l_m starting batches in σ_A. Note that $l_m \ge F(1, m+1)$ by the definition of l_m. Now we prove that σ_A^1 is not an α-nice schedule. Suppose to the contrary that σ_A^1 is an α-nice schedule. We claim that the following statements hold for σ_A^1:

- $B'_1(A), \ldots, B'_m(A)$ exist and $S'_i(A) \le (1+\alpha)^{m-i+2}(S'_{i-1}(A) + 1) - 1$ for $i = 1, \ldots, m$,
- at time $S'_m(A)$ the m machines are occupied by the m restricted batches, and
- each $B'_i(A)$ $(1 \le i \le m)$ is just $B_{j_i}(A)$ for some $j_i \le F(m-i+2, m+1)$.

To this end, we recall that $(1+x)^{m+1} - (1+x) - 1$ is monotonically increasing in $x > 0$. Then we have

$$\begin{aligned}
&S_{m+1}(A) - \big(S_1(A) + 1\big) \\
&\le \big(1+\alpha\big)^m\big(S_1(A) + 1\big) - \big(S_1(A) + 1\big) - 1 \\
&\le \big(1+\alpha\big)^{m+1} - \big(1+\alpha\big) - 1
\end{aligned}$$

$$
\begin{aligned}
&< \big(1+\beta_m\big)^{m+1} - \big(1+\beta_m\big) - 1 \\
&< \big(1+\alpha(m+1,1)\big)^{m+1} - \big(1+\alpha(m+1,1)\big) - 1 \\
&= 0,
\end{aligned}
$$

where the first two inequalities follow from the assumption that σ_A^1 is an α-nice schedule, and the rest inequalities hold as $0 < \alpha < \beta_m \le \alpha(2m+1, 2) < \alpha(m+1, 1)$ from (1) and (2). This implies that $S_{m+1}(A) < S_1(A) + 1$, and so, the first $m+1$ batches in σ_A^1 must contain a restricted batch. Hence, $B'_1(A)$ exists and $S'_1(A) \le S_{m+1}(A) \le (1+\alpha)^{m+1} - 1 = (1+\alpha)^{m+1}(S'_0(A)+1) - 1 = (1+\alpha)^{F(m-1+2,m+1)}(S'_0(A)+1) - 1$. This further implies that $B'_1(A)$ is just $B_{j_1}(A)$ for some $j_1 \le m+1 = F(m-1+2, m+1)$.

Let $k \le m$ be the maximum so that $B'_1(A), \ldots, B'_{k-1}(A)$ exist and, at time $S'_{k-1}(A)$, $k-1$ machines are occupied by the $k-1$ restricted batches. For $i = 1, \ldots, k-1$, as there are at most $m-i+1$ free batches between $B'_{i-1}(A)$ and $B'_i(A)$, each $B'_i(A)$ $(1 \le i \le k-1)$ is just $B_{j_i}(A)$ for some $j_i \le m-i+2+j_{i-1} \le F(m-i+2, m+1)$ and $S'_i(A) \le (1+\alpha)^{m-i+2}(S'_{i-1}(A)+1) - 1$.

From the fact that $B_{m-k+2,k-1}(A)$ is some batch $B_j(A)$ with $j = m-k+2+j_{k-1} \le m-k+2+F(m-(k-1)+2, m+1) = F(m-k+2, m+1) < F(1, m+1) \le l_m$, we have

$$
\begin{aligned}
&S_{m-k+2,k-1}(A) - \big(S'_1(A)+1\big) \\
&\le \big(1+\alpha\big)^{m-k+2}\big(S'_{k-1}(A)+1\big) - 1 - \big(S'_1(A)+1\big) \\
&\le \big(1+\alpha\big)^{F\left(m-k+2,m\right)}\big(S'_1(A)+1\big) - \big(S'_1(A)+1\big) - 1 \\
&\le \big(1+\alpha\big)^{F\left(m-k+2,m+1\right)} - \big(1+\alpha\big)^{m+1} - 1 \\
&< \big(1+\alpha\big)^{F\left(1,m+1\right)} - \big(1+\alpha\big)^{m+1} - 1 \\
&< \big(1+\beta_m\big)^{F\left(1,m+1\right)} - \big(1+\beta_m\big)^{m+1} - 1 \\
&= 0,
\end{aligned}
$$

where the first inequality follows from the assumption that σ_A^1 is an α-nice schedule, the next two inequalities follow from the fact that $S'_i(A) \le (1+\alpha)^{m-i+2}(S'_{i-1}(A)+1) - 1$ for $i = 1, \ldots, k-1$, and the last inequality holds as $0 < \alpha < \beta_m$. This implies that $S_{m-k+2,k-1}(A) < S'_1(A) + 1$.

Suppose to the contrary that all $B_{i,k-1}(A)$, $1 \le i \le m-k+2$, are free batches. Since there are only $m-k+1$ idle machines at time $S'_{k-1}(A)$, we have $S_{m-k+2,k-1}(A) \ge S'_1(A) + 1$, a contradiction. So some batch in $\{B_{i,k-1}(A), 1 \le i \le m-k+2\}$ is a restricted batch. Let $B'_k(A)$ be the first starting restricted batch in $\{B_{i,k-1}(A) : 1 \le i \le m-k+2\}$. Then $B'_k(A)$ is just $B_{j_k}(A)$ for some $j_k \le m-k+2+j_{k-1} \le F(m-k+2, m+1)$. Therefore, $S'_k(A) \le S_{m-k+2,k-1}(A) \le (1+\alpha)^{m-k+2}(S'_{k-1}(A)+1) - 1$ and $S'_k(A) \le S_{m-k+2,k-1}(A) < S'_1(A) + 1$. The claim follows.

To continue the proof, we assume that, at time $S'_m(A)$, a new job arrives. Note that

$$\begin{aligned}
&\big(1+\alpha\big)S'_m(A)+\alpha-\big(S'_1(A)+1\big)\\
&\quad\le \big(1+\alpha\big)^{F(1,m)}\big(S'_1(A)+1\big)-\big(S'_1(A)+1\big)-1\\
&\quad\le \big(1+\alpha\big)^{F(1,m+1)}-\big(1+\alpha\big)^{m+1}-1\\
&\quad< \big(1+\beta_m\big)^{F(1,m+1)}-\big(1+\beta_m\big)^{m+1}-1\\
&\quad=0,
\end{aligned}$$

where the first two inequalities follow from the fact that $S'_i(A)\le(1+\alpha)^{m-i+2}(S'_{i-1}(A)+1)-1$ for $i=1,\ldots,m$, and the last inequality holds as $0<\alpha<\beta_m$. Then, from the above claim, at time $(1+\alpha)S'_m(A)+\alpha$, each of $B'_i(A)$, $1\le i\le m$, has not been completed. So $S_{1,m}(A)\ge S'_1(A)+1>(1+\alpha)S'_m(A)+\alpha$. Note that $B_{1,m}(A)$ is $B_{j'}(A)$ with $j'=j_m+1\le 1+F(2,m+1)=F(1,m+1)\le l_m$. Then σ^1_A is not an α-nice schedule, a contradiction.

The above discussion implies that σ^1_A (and so, σ_A) is not an α-nice schedule. Then σ_α, which is just the schedule $\sigma^1_{\text{LAZY}(\alpha)}$, is not an α-nice schedule. The result follows. □

Lemma 2.3 *If σ_α is an α-regular schedule, then $\sigma_{\text{LAZY}(\alpha)}$ is also α-regular.*

Proof Assume that σ_α is an α-regular schedule. The definitions of σ_α and $\sigma_{\text{LAZY}(\alpha)}$ imply that the batches $B_i(\text{LAZY}(\alpha))=B_i(\alpha)$, $1\le i\le l_m$, are α-regular in $\sigma_{\text{LAZY}(\alpha)}$. From Lemma 2.2, we have $\alpha\ge\beta_m$. For each $i>l_m$, we have $(1+\alpha)S_{i-1}(\text{LAZY}(\alpha))+\alpha-(S_{i-1}(\text{LAZY}(\alpha))+1)=\alpha(S_{i-1}(\text{LAZY}(\alpha))+1)-1\ge\alpha(1+\alpha)^{i-1}-1\ge\beta_m(1+\beta_m)^{l_m}-1\ge 0$. Thus $B_{i-1}(\text{LAZY}(\alpha))$ completes by time $(1+\alpha)S_{i-1}(\text{LAZY}(\alpha))+\alpha$. Consequently, $B_i(\text{LAZY}(\alpha))$ is a free and α-regular batch in $\sigma_{\text{LAZY}(\alpha)}$, as required. Then $\sigma_{\text{LAZY}(\alpha)}$ is α-regular. The result follows. □

3 The lower bound

Lemma 3.1 *Let α be a real number with $0<\alpha<1$. If there exists an algorithm A' such that $\sigma_{A'}$ is an α-nice schedule, then σ_α is an α-regular schedule.*

Proof Assume that $\sigma_{A'}$ is an α-nice schedule with sufficiently many batches. Then $S_i(A')\le(1+\alpha)S_{i-1}(A')+\alpha$ for $i\ge 1$. From Lemma 2.2, we have $\alpha\ge\beta_m$. If $\alpha\ge\alpha(2m+1,2)$, the result follows from Lemma 2.1iii. Hence we assume in the following that $\beta_m\le\alpha<\alpha(2m+1,2)$. Suppose to the contrary that σ_α is not an α-regular schedule. Set $n'=\min\{i:1\le i\le l_m, S_i(\alpha)>(1+\alpha)S_{i-1}(\alpha)+\alpha\}$.

Based on $\sigma_{A'}$, we define a new schedule $\sigma^*_{A'}$ which consists of n' batches and has the same batch sequence on each machine as $\sigma_{A'}$. The starting time of each batch $B^*_i(A')$ in $\sigma^*_{A'}$ is given by $S^*_i(A')=(1+\alpha)^i-1$ and, furthermore, $B^*_i(A')$ is (assigned) restricted in $\sigma^*_{A'}$ if and only if $B_i(A')$ is restricted in $\sigma_{A'}$. Then $S^*_i(A')=(1+\alpha)S^*_{i-1}(A')+\alpha$ for $1\le i\le n'$. Since $S_i(A')\le(1+\alpha)S_{i-1}(A')+\alpha$ for $i\ge 1$, it can be verified that for every batch indexes i and j with $1\le i<j\le n'$, we have $S^*_j(A')-S^*_i(A')\ge S_j(A')-S_i(A')$. It follows that $\sigma^*_{A'}$ is also a feasible schedule. We also define a new job

instance $\mathcal{I}'(\alpha)$ which consists of the first n' jobs in $\mathcal{I}(\alpha)$. Note that $\sigma^*_{A'}$ can be taken as a feasible schedule for instance $\mathcal{I}'(\alpha)$ in which all batches are α-regular. Hence there exists an α-regular schedule for instance $\mathcal{I}'(\alpha)$.

It can be observed that, for every α-regular schedule for instance $\mathcal{I}'(\alpha)$, the first batch is exactly $B_1(\alpha)$.

Let π be an α-regular schedule for instance $\mathcal{I}'(\alpha)$. The i-th batch in π is denoted by B_i^π. Define $e(\pi) = \max\{k : 1 \le k \le n',\ B_i^\pi = B_i(\alpha) \text{ for } i \le k\}$, where we overload the notation "$B_i^\pi = B_i(\alpha)$" to indicate that the two batches include the same jobs, have the same starting time and are both free batches or both restricted batches. Then $e(\pi) \ge 1$. We can choose π so that $e(\pi)$ is as large as possible. Since π is an α-regular schedule, we have $S_i(\alpha) = S_i^\pi = (1+\alpha)^i - 1$ for $1 \le i \le e(\pi)$. If $e(\pi) = n'$, then $B_i(\alpha)$ $(1 \le i \le n')$ are all α-regular, a contradiction. So in the following we suppose that $e(\pi) < n'$. Then $B_i^\pi = B_i(\alpha)$ for $1 \le i \le e(\pi)$ and $B^\pi_{e(\pi)+1} \ne B_{e(\pi)+1}(\alpha)$. From the fact that π is an α-regular schedule and by the implementation of algorithm LAZY(α), $B_{e(\pi)+1}(\alpha)$ is α-regular, and so, $S_{e(\pi)+1}(\alpha) = S^\pi_{e(\pi)+1} = (1+\alpha)^{e(\pi)+1} - 1$.

If $B^\pi_{e(\pi)} = B_{e(\pi)}(\alpha)$ is a restricted batch, then both $B^\pi_{e(\pi)+1}$ and $B_{e(\pi)+1}(\alpha)$ are free batches consisting of only one job $J_{e(\pi)+1}(\alpha)$. But then $B^\pi_{e(\pi)+1} = B_{e(\pi)+1}(\alpha)$, a contradiction. Hence $B^\pi_{e(\pi)} = B_{e(\pi)}(\alpha)$ is a free batch.

If $B^\pi_{e(\pi)+1}$ and $B_{e(\pi)+1}(\alpha)$ are both restricted batches, then both of them consist of the same jobs. Then we have $B^\pi_{e(\pi)+1} = B_{e(\pi)+1}(\alpha)$, a contradiction.

If $B^\pi_{e(\pi)+1}$ is a free batch, then $B^\pi_{e(\pi)+1} = \{J_{e(\pi)+1}(\alpha)\}$ and at time $(1+\alpha)S^\pi_{e(\pi)} + \alpha = (1+\alpha)S_{e(\pi)}(\alpha) + \alpha$, some machine is idle. The implementation of algorithm LAZY(α) implies that $B_{e(\pi)+1}(\alpha) = \{J_{e(\pi)+1}(\alpha)\}$ is also a free batch. Thus, $B^\pi_{e(\pi)+1} = B_{e(\pi)+1}(\alpha)$, a contradiction.

Hence, the only possibility is that $B^\pi_{e(\pi)+1}$ is a restricted batch and $B_{e(\pi)+1}(\alpha)$ is a free batch. Noticing that $B^\pi_{e(\pi)} = B_{e(\pi)}(\alpha)$ is a free batch, the implementation of algorithm LAZY(α) implies that $B_{e(\pi)+1}(\alpha) = \{J_{e(\pi)+1}(\alpha)\}$ is also an α-regular batch. So in the case $e(\pi) = n' - 1$, $B_{n'}(\alpha)$ is α-regular, a contradiction.

Suppose that $e(\pi) \le n' - 2$. We obtain a new schedule π' from π so that $B_i^{\pi'} = B_i^\pi$ for $1 \le i \le e(\pi)$, $B^{\pi'}_{e(\pi)+1} = \{J_{e(\pi)+1}(\alpha)\}$, and, for each $e(\pi)+2 \le i \le n'$, $B_i^{\pi'}$ is (assigned) restricted in π' if and only if B^π_{i-1} is restricted in π. The α-regularity of π' can be observed from the fact that we can always schedule each batch $B_i^{\pi'}$ with $e(\pi)+2 \le i \le n'$ (no matter restricted or free) on the machine occupied by B^π_{i-1} in π. Now π' is α-regular and $B_i^{\pi'} = B_i(\alpha)$ for $1 \le i \le e(\pi)+1$. This contradicts the choice of π. The result follows. □

Lemma 3.2 *Suppose that* $0 < \alpha < \beta < 1$. *If* σ_α *is* α*-regular, then* σ_β *is* β*-regular.*

Proof If σ_α is α-regular, then $\sigma_{\text{LAZY}(\alpha)}$ is α-regular by Lemma 2.3. Since $\alpha < \beta$, $\sigma_{\text{LAZY}(\alpha)}$ is β-nice. By Lemma 3.1, σ_β is β-regular. The result follows. □

We define

$$\alpha_m = \min\left\{\alpha(s,t) : 1 \le t < s \le l_m,\ \sigma_{\alpha(s,t)} \text{ is } \alpha(s,t)\text{-regular}\right\}. \tag{3}$$

From Lemmas 2.1iii and 2.2, α_m is well defined in (3) and

$$\beta_m \le \alpha_m \le \alpha(2m+1, 2). \tag{4}$$

Lemma 3.3 $\alpha_m = \min\{\alpha : 0 < \alpha < 1,\ \sigma_\alpha$ *is* α*-regular*$\}$.

Proof Write $\gamma = \min\{\alpha : 0 < \alpha < 1,\ \sigma_\alpha$ is α-regular$\}$. From Lemma 2.2 we have $\gamma \ge \beta_m$ and by the definition of α_m, we have $\gamma \le \alpha_m$. We first prove that $\gamma \in \{\alpha(s,t) : 1 \le t < s \le l_m\}$.

Suppose to the contrary that $\gamma \notin \{\alpha(s,t) : 1 \le t < s \le l_m\}$. Then $\gamma < \alpha_m$ by the definition of α_m and γ. Furthermore, the assumption also implies the following Claim 1.

Claim 1 *For each pair of positive integers s and t with* $1 \le t < s \le l_m$*, we have* $(1+\gamma)^s - (1+\gamma)^t - 1 \ne 0$.

Since s and t have finitely many choices, from Claim 1, we have the following Claim 2.

Claim 2 *There is a positive number* $\alpha < \gamma$ *such that, for each pair of positive integers s and t with* $1 \le t < s \le l_m$, $(1+\gamma)^s - (1+\gamma)^t - 1 > 0$ *if and only if* $(1+\alpha)^s - (1+\alpha)^t - 1 > 0$.

Let α be given as in Claim 2. Note that the definition of γ implies that σ_γ is a γ-regular schedule. We first show that σ_α is also an α-regular schedule.

We construct a new schedule σ^* from σ_γ by the following way. The batches in σ^* are given by $B_1^*, B_2^*, \ldots, B_{l_m}^*$ so that B_i^* is (assigned) restricted in σ^* if and only if $B_i(\gamma)$ is restricted in σ_γ for each i with $1 \le i \le l_m$. Furthermore, each batch B_i^* is processed on the machine occupied by $B_i(\gamma)$ with starting time $S_i^* = (1+\alpha)^i - 1$ in σ^*. We only need to show that such a schedule σ^* is feasible, or equivalently, there is no violated batch in σ^*. Here, a restricted batch B_i^* is called violated if B_i^* overlaps with some restricted batch $B_{i'}^*$ with $i' < i$ on the same machine in σ^*, and a free batch B_i^* is called violated if B_i^* overlaps with some batch $B_{i'}^*$ with $i' < i$ on the same machine in σ^*.

If possible let B_s^* be the first violated batch in σ^*. Then there is a batch B_t^* with $t < s$ on the same machine such that B_s^* overlaps with B_t^* in σ^* but $B_s(\gamma)$ processed after $B_t(\gamma)$ in σ_γ. Then we have $S_t^* + 1 > S_s^*$ and $S_t(\gamma) + 1 \le S_s(\gamma)$, and therefore, $(1+\alpha)^t > (1+\alpha)^s - 1$ and $(1+\gamma)^t \le (1+\gamma)^s - 1$. From Claim 1, we have $(1+\gamma)^s > (1+\gamma)^t + 1$. Then the facts that $(1+\alpha)^s < (1+\alpha)^t + 1$ and $(1+\gamma)^s > (1+\gamma)^t + 1$ contradict Claim 2.

The above discussion implies that $\gamma \in \{\alpha(s,t) : 1 \le t < s \le l_m\}$. Then the only possibility is that $\gamma = \alpha_m$. The result follows. □

Theorem 3.4 *There exists no online algorithm with a competitive ratio of less than* $1 + \alpha_m$.

Proof Suppose to the contrary that A is an online algorithm with a competitive ratio of $1+\gamma$, where $\gamma < \alpha_m$. Then in schedule σ_A with sufficiently many batches, $S_i(A) \le (1+\gamma)S_{i-1}(A) + \gamma$ for each $i \ge 1$, where $S_0(A) = 0$. Thus σ_A is a γ-nice schedule. From Lemma 3.1, σ_γ is a γ-regular schedule. This contradicts Lemma 3.3. The result follows. □

4 An online algorithm

The following online algorithm ALG_m is closely related to the α_m-regular schedule $\sigma_{\mathrm{LAZY}(\alpha_m)}$ for instance $\mathcal{I}(\mathrm{LAZY}(\alpha_m))$. The intuition of the algorithm ALG_m can be stated as follows.

According to the arriving of the jobs, we generate a sequence of positive integers $j_1, j_2, \ldots, j_n$ so that $j_1 < j_2 < \cdots < j_n$. When the first job arrives at time 0, let $j_1 = 1$. When the first job arrives at a time $t_1 > 0$, we determine j_1 so that $t_1 \in ((1+\alpha_m)^{j_1-1} - 1, (1+\alpha_m)^{j_1} - 1]$. Then we start the first batch B_1^{ALG} as a free batch at time $S_1^{\mathrm{ALG}} = (1+\alpha_m)^{j_1} - 1$. Generally, suppose that the first $i-1$ batches $B_1^{\mathrm{ALG}}, B_2^{\mathrm{ALG}}, \ldots, B_{i-1}^{\mathrm{ALG}}$ have been generated and a new job arrives at a time $t_i > S_{i-1}^{\mathrm{ALG}}$. Then we determine j_i so that $t_i \in ((1+\alpha_m)^{j_i-1} - 1, (1+\alpha_m)^{j_i} - 1]$ and generate a batch B_i^{ALG} with starting time $S_i^{\mathrm{ALG}} = (1+\alpha_m)^{j_i} - 1$ in ALG_m. Thus we have $S_i^{\mathrm{ALG}} = (1+\alpha_m)^{j_i} - 1$ for each i. We take the convention that B_i^{ALG} is an assigned restricted batch in ALG_m if and only if $B_i(\mathrm{LAZY}(\alpha_m))$ is a restricted batch in $\sigma_{\mathrm{LAZY}(\alpha_m)}$. Note that in algorithm ALG_m, an assigned restricted batch can not be restarted again.

Algorithm ALG_m:

1. Set $i := 0$.
2. Wait until $U(t) \neq \emptyset$.
3. Set $i := i+1$. Let j_i be the minimum positive integer so that $t \leq (1+\alpha_m)^{j_i} - 1$ and wait until the current time $t = (1+\alpha_m)^{j_i} - 1$.
4. If $B_i(\mathrm{LAZY}(\alpha_m))$ is a restricted batch in $\sigma_{\mathrm{LAZY}(\alpha_m)}$, interrupt all free batches at time t. B_i^{ALG} will be an assigned restricted batch consisting of all interrupted jobs and all jobs in $U(t)$.
5. If $B_i(\mathrm{LAZY}(\alpha_m))$ is a free batch in $\sigma_{\mathrm{LAZY}(\alpha_m)}$, set $B_i^{\mathrm{ALG}} = U(t)$ and take B_i^{ALG} as a free batch.
6. Start to process B_i^{ALG} at time $S_i^{\mathrm{ALG}} = t$ on the machine of $B_i(\mathrm{LAZY}(\alpha_m))$ in $\sigma_{\mathrm{LAZY}(\alpha_m)}$.
7. Go to step 2. □

Algorithm ALG_m runs in linear time, since we only take action when a new job arrives.

Theorem 4.1 *Algorithm* ALG_m *has a competitive ratio at most* $1+\alpha_m$.

Proof Let $\mathcal{J}$ be an arbitrary job instance. Let σ^{ALG} be the schedule generated by algorithm ALG_m for instance $\mathcal{J}$. Let $B_1^{\mathrm{ALG}}, B_2^{\mathrm{ALG}}, \ldots, B_n^{\mathrm{ALG}}$ be the starting batches in σ^{ALG}, where each B_i^{ALG} has a starting time S_i^{ALG}. We first show that σ^{ALG} is feasible (no overlapping batches, and no interruptions of restricted batches).

By the implementation of ALG_m, for $1 \leq i \leq n$, B_i^{ALG} is processed on the machine occupied by $B_i(\mathrm{LAZY}(\alpha_m))$ in $\sigma_{\mathrm{LAZY}(\alpha_m)}$ with $S_i^{\mathrm{ALG}} = (1+\alpha_m)^{j_i} - 1$. Moreover, B_i^{ALG} is an assigned restricted batch in σ^{ALG} if and only if $B_i(\mathrm{LAZY}(\alpha_m))$ is a restricted batch in $\sigma_{\mathrm{LAZY}(\alpha_m)}$. Let i and k be any two integers with $1 \leq i < k \leq n$. As there are

$k - i$ batches between B_i^{ALG} and B_k^{ALG} in σ^{ALG}, we have $j_k - j_i \geq k - i$. Thus,

$$\begin{aligned} & S_k^{\mathrm{ALG}} - S_i^{\mathrm{ALG}} \\ & = \left(1+\alpha_m\right)^{j_k} - \left(1+\alpha_m\right)^{j_i} \\ & = \left(1+\alpha_m\right)^{j_i}\left((1+\alpha_m)^{j_k-j_i} - 1\right) \\ & \geq \left(1+\alpha_m\right)^{i}\left((1+\alpha_m)^{k-i} - 1\right) \\ & = S_k\left(\mathrm{LAZY}(\alpha_m)\right) - S_i\left(\mathrm{LAZY}(\alpha_m)\right). \end{aligned}$$

It follows that σ^{ALG} is feasible.

Furthermore, it is clear that each job is started (for the first time) early enough so that it can complete in time to maintain the competitive ratio $1 + \alpha_m$. The result follows. □

From Theorems 3.4 and 4.1, we conclude the main result of this paper as follows.

Theorem 4.2 *Algorithm* ALG$_m$ *is a best possible online algorithm for problem* $Pm|online,\ r_j,\ p_j = 1,\ p\text{-}batch,\ b = \infty,\ L\text{-}restart|C_{\max}$. □

As we can see, α_m is not presented as an explicit formulation of m in this paper. We list some values of α_m and the competitive ratio $1 + \alpha_m$ in the following table.

m	$\alpha_m = \alpha(i, j)$	Competitive ratio
2	$\alpha(6, 3) \approx 0.1740$	1.1740
3	$\alpha(10, 4) \approx 0.0926$	1.0926
4	$\alpha(17, 9) \approx 0.0599$	1.0599
5	$\alpha(20, 6) \approx 0.0421$	1.0421
6	$\alpha(30, 13) \approx 0.0308$	1.0308
7	$\alpha(44, 26) \approx 0.0242$	1.0242
8	$\alpha(46, 17) \approx 0.0190$	1.0190
9	$\alpha(60, 27) \approx 0.0155$	1.0155
10	$\alpha(60, 11) \approx 0.0128$	1.0128
11	$\alpha(97, 57) \approx 0.0109$	1.0109
12	$\alpha(100, 46) \approx 0.0093$	1.0093
13	$\alpha(101, 27) \approx 0.0080$	1.0080
14	$\alpha(122, 42) \approx 0.0070$	1.0070

Acknowledgments The authors would like to thank the associate editor and two anonymous referees for their constructive comments and kind suggestions. This work was supported by NSFC (11271338), NSFC (11171313), NSFC (11401604), NSFHN (132300410392), and Henan Institute of Engineering (Y2010018).

References

Chen H, Zhang YP, Yong XR (2009) On-line scheduling on a single bounded batch processing machine with restarts. In: Proceedings of the first international workshop on education technology and computer science. IEEE Computer Society, Washington, pp 868–871

Deng XT, Poon CK, Zhang YZ (2003) Approximation algorithms in batch processing. J Combin Optim 7:247–257

Epstein L, Van Stee R (2003) Lower bounds for on-line single-machine scheduling. Theoret Comput Sci 299:439–450

Fu RY, Tian J, Yuan JJ, Lin YX (2007) On-line scheduling in a parallel batch processing system to minimize makespan using restarts. Theoret Comput Sci 374:196–202

Fu RY, Tian J, Yuan JJ, He C (2008) On-line scheduling on a batch machine to minimize makespan with limited restarts. Oper Res Lett 36:255–258

Fu RY, Cheng TCE, Ng CT, Yuan JJ (2010) Online scheduling on two parallel-batching machines with limited restarts to minimize the makespan. Inform Process Lett 110:444–450

Hoogeveen H, Potts CN, Woeginger GJ (2000) On-line scheduling on a single machine: maximizing the number of early jobs. Oper Res Lett 27:193–197

Hoogeveen JA, Vestjens APA (2000) A best possible deterministic on-line algorithm for minimizing maximum delivery time on a single machine. SIAM J Discret Math 13:56–63

Liu PH, Lu XW, Fang Y (2012) A best possible deterministic on-line algorithm for minimizing makespan on parallel batch machines. J Sched 15:77–81

Liu HL, Yuan JJ (2014) Online scheduling of equal length jobs on a bounded parallel batch machine with restart or limited restart. Theoret Comput Sci 543:24–36

Nong QQ, Cheng TCE, Ng CT (2008) An improved on-line algorithm for scheduling on two unrestrictive parallel-batch processing machines. Oper Res Lett 36:584–588

Poon CK, Yu WC (2005) Online scheduling algorithms for a batch machine with finite capacity. J Combin Optim 9:167–186

Tian J, Cheng TCE, Ng CT, Yuan JJ (2009a) Online scheduling on unbounded parallel-batch machines to minimize the makespan. Inform Process Lett 109:1211–1215

Tian J, Fu RY, Yuan JJ (2009b) A best online algorithm for scheduling on two parallel batch machines. Theoret Comput Sci 410:2291–2294

Tian J, Fu RY, Yuan JJ (2014) Online over time scheduling on parallel-batch machines: a survey. J Oper Res Soc China 2:445–454

Uzsoy R, Lee CY, Martin-Vega LA (1992) A review of production planning and scheduling models in the semiconductor industry, part I: system characteristics, performance evaluation and production planning. IIE Trans Sched Logist 24:47–61

Uzsoy R, Lee CY, Martin-Vega LA (1994) A survey of production planning and scheduling models in the semiconductor industry, part II: shop-floor control. IIE Trans Sched Logist 26:44–55

Van den Akker M, Hoogeveen H, Vakhania N (2003) Restarts can help in the online minimization of the maximum delivery time on a single machine. J Sched 3:333–341

Van stee R, La Poutré H (2005) Minimizing the total completion time on-line on a single machine using restarts. J Algorithms 57:95–129

Yuan JJ, Fu RY, Ng CT, Cheng TCE (2011) A best online algorithm for unbounded parallel-batch scheduling with restarts to minimize makespan. J Sched 14:361–369

Zhang GC, Cai XQ, Wong CK (2001) On-line algorithms for minimizing makespan on batch processing machines. Naval Res Logist 48:241–258

Zhang GC, Cai XQ, Wong CK (2003) Optimal on-line algorithms for scheduling on parallel batch processing machines. IIE Trans 35:175–181