J Sched (2012) 15:77–81
DOI 10.1007/s10951-009-0154-4

A best possible deterministic on-line algorithm for minimizing makespan on parallel batch machines

Peihai Liu · Xiwen Lu · Yang Fang

Published online: 31 December 2009

Abstract We study on-line scheduling on parallel batch machines. Jobs arrive over time. A batch processing machine can handle up to B jobs simultaneously. The jobs that are processed together form a batch and all jobs in a batch start and are completed at the same time. The processing time of a batch is given by the processing time of the longest job in the batch. The objective is to minimize the makespan. We deal with the unbounded model, where B is sufficiently large. We first show that no deterministic on-line algorithm can have a competitive ratio of less than $1 + (\sqrt{m^2+4} - m)/2$, where m is the number of parallel batch machines. We then present an on-line algorithm which is the one best possible for any specific values of m.

Keywords Scheduling · Parallel Batch Machines · On-line

1 Introduction

We consider the problem of on-line scheduling on m parallel batch processing machines. A parallel batch processing machine is modeled as a system that can handle up to B jobs simultaneously as a batch. The processing time of a batch is the time required for processing the longest job in the batch, and all the jobs in a batch start and are completed at the same time. A set of n independent jobs are given. Each job J_j $(1 \le j \le n)$ becomes available at its release time r_j, which is not known in advance, and its processing time p_j only becomes known at its arrival. The problem involves assigning all the jobs to batches and machines and determining the starting times of the resulting batches in such a way that the makespan, i.e., $\max_{1\le j\le n} C_j$, is minimized, where C_j is the completion time of job J_j. In the scheduling notation introduced by Graham et al. (1979), this model is expressed as $P|r_j, B = \infty, \text{on-line}|C_{\max}$.

Scheduling of batch processing machines has been extensively studied in the last decade (Brucker et al. 1998; Chen et al. 2001, 2004; Lee et al. 1992, 2005; Ng et al. 2002, 2003; Uzsoy 1994; Yuan et al. 2004). According to the limit on the size of each batch, there are two distinct models. One is the restrictive model in which the bound B on each batch size is finite, i.e., $B < n$. Problems of this model are motivated by the burn-in operations in semiconductor manufacturing, in which a batch of integrated circuits are placed in an oven and then exposed to a high temperature. Each circuit has a pre-specified minimum burn-in time and the burn-in oven has a limited capacity. The other is the unrestrictive model, in which there is effectively no limit on the sizes of batches, i.e., $B = \infty$. Scheduling problems of this model arise in situations where compositions need to be hardened in kilns and the kiln is sufficiently large, so that it does not restrict the batch sizes.

For many on-line scheduling problems, because of a lack of information, it is normally notpossible to have on-line algorithms that guarantee optimal solutions to be delivered. Researchers therefore turn to studying approximate on-line algorithms for this kind of problem. The quality of an on-line algorithm is typically assessed by its competitive ratio: the nearer the ratio is to 1, the better the algorithm is. We say that an algorithm is ρ-competitive if for any input instance, it always returns a feasible solution with an objective value not greater than ρ times the optimal (off-line) solution.

Let us survey the previous related results. Lee and Uzsoy (1999) provided a number of heuristics for the problem $1|r_j, B < n|C_{\max}$. Liu and Yu (2000) proved that the

P. Liu · X. Lu (✉) · Y. Fang
School of Science, East China University of Science and Technology, Shanghai 200237, People's Republic of China
e-mail: xwlu@ecust.edu.cn

problem is NP-hard even if there are only two release dates, and they derived a pseudo-polynomial time algorithm for the case where the number of release dates is fixed. Zhang et al. (2001) considered the on-line version of the problem. They dealt with both the unrestrictive and restrictive models. For the unrestrictive model, they derived an optimal on-line algorithm with a competitive ratio of $(\sqrt{5}+1)/2$. For the restrictive model, they provided a 2-competitive on-line algorithm. In the same paper, they considered the problem $Pm|B=\infty, r_j, \text{on-line}|C_{\max}$ and developed an $(1+\beta_m)$-competitive on-line algorithm, where m is the number of machines, and $0<\beta_m<1$ is a solution of the equation $\beta_m=(1-\beta_m)^{m-1}$. Zhang et al. (2003) addressed the problems $P|B<n, r_j, p_j=p, \text{on-line}|C_{\max}$ and $P|B=\infty, r_j, p_j=p, \text{on-line}|C_{\max}$. In the two problems, there is an assumption that the processing times of the jobs to be scheduled are identical. They proved that there is no on-line algorithm with a competitive ratio smaller than $(\sqrt{5}+1)/2$ for $P|B<n, r_j, p_j=p, \text{on-line}|C_{\max}$ and they also presented an on-line algorithm with a competitive ratio matching the lower bound. For $Pm|B=\infty, r_j, p_j=p, \text{on-line}|C_{\max}$, they provided a best possible on-line algorithm with a competitive ratio of $1+\gamma_m$, where $0<\gamma_m<1$ is a solution of the equation $(1+\gamma_m)^{m+1}=\gamma_m+2$. Nong et al. (2008) considered the problem $P2|B=\infty, r_j, \text{on-line}|C_{\max}$. They gave an on-line algorithm which is $\sqrt{2}$-competitive. Tian et al. (2009) showed that there cannot exist a deterministic on-line algorithm with a better competitive ratio for the two parallel machines case.

In this paper, we consider the problem $Pm|B=\infty, r_j, \text{on-line}|C_{\max}$. We first provide an on-line algorithm with a competitive ratio of $1+\alpha_m$, where $\alpha_m=(\sqrt{m^2+4}-m)/2$. We then show that no algorithm can have a competitive ratio of less than $1+\alpha_m$, where α_m is such that $\alpha_m^2=1-m\alpha_m$. Thus our algorithm is the one best possible.

2 An on-line algorithm

Let $J(t)$ be the set of the jobs which are available but not yet scheduled at time t. Denote by $p(t)$ the largest processing time of the jobs in $J(t)$. Denote by $r(t)$ the smallest release time of the jobs in $J(t)$. Let $\alpha=(\sqrt{m^2+4}-m)/2$; we give an on-line algorithm for $Pm|r_j, B=\infty, \text{on-line}|C_{\max}$ as follows.

Algorithm $H_m^{\infty}(\alpha)$ At time t, if a machine is idle and there are jobs available but not yet scheduled and $t\geq(1+\alpha)r(t)+\alpha p(t)$, then start all the available jobs as a single batch on the idle machine. Otherwise, do nothing, but wait.

Now, we analyze the competitive ratio of this algorithm.

Given an instance I, denote by σ the schedule generated by Algorithm $H_m^{\infty}(\alpha)$ and by π the optimal schedule. For a schedule χ, let $C_{\max}(\chi)$ denote its makespan and let $S_j(\chi)$ denote the starting time of job J_j in the schedule χ. For any two jobs J_i and J_j in two different batches in σ, if $S_i(\sigma)>S_j(\sigma)$, then all the jobs in the batch to which job J_i is assigned are released after $S_j(\sigma)$ by the description of Algorithm $H_m^{\infty}(\alpha)$. This implies that (1) $S_i(\sigma)>(1+\alpha)S_j(\sigma)+\alpha p_i$ and (2) $C_{\max}(\pi)>S_j(\sigma)+p_i$. For a batch starting at time s in σ, we say that the batch is regular if it starts at time $(1+\alpha)r(s)+\alpha p(s)$. Clearly, if the batch is not regular, then $s>(1+\alpha)r(s)+\alpha p(s)$, which means that the m machines are processing jobs during the interval $[r(s), s]$.

We will prove that $H_m^{\infty}(\alpha)$ has a competitive ratio of $1+\alpha$. In our proof, we work with the smallest counterexample, where size is measured in terms of the number of jobs. Let I be such a smallest counterexample, and let σ be the schedule created by $H_m^{\infty}(\alpha)$ for I. To prove that such a counterexample cannot exist, we start by proving four structure properties that the schedule σ for an alleged smallest counterexample must satisfy in the following four lemmas.

We assume there are n batches in σ. We denote by B_i the batch i.

Lemma 1 *In the schedule σ for the alleged smallest counterexample I, there is only one job in each batch.*

Proof Suppose to the contrary that there is more than one job in some batch of σ; then there are at least $n+1$ jobs in the instance I. We can find a counterexample I' consisting of n jobs, which contradicts our choice of I. For each batch B_i of σ, we define a new job J_i for the instance I', $r_i=\min_{J_j\in B_i} r_j$, $p_i=\max_{J_j\in B_i} p_j$. Apply $H_m^{\infty}(\alpha)$ to I'. Then we obtain a schedule σ' for I' that is identical to σ in the sense that the processing times and the starting times of the batches in the schedule σ' are the same as those in σ. Thus, the makespan of the resulting schedule is not smaller than $C_{\max}(\sigma)$. On the other hand, it is evident that the makespan of the optimal schedule π' for I' is not greater than $C_{\max}(\pi)$. Therefore, $C_{\max}(\sigma')/C_{\max}(\pi')\geq C_{\max}(\sigma)/C_{\max}(\pi)>1+\alpha$. □

From Lemma 1 we know that there is only one job in each batch of σ. Without loss of generality, we assume that the job in B_i is J_i. Thus the release time, processing time, starting time and completion time of B_i in σ can be denoted by r_i, p_i, $S_i(\sigma)$ and $C_i(\sigma)$. For convenience, we index the jobs in σ in the order such that $S_1(\sigma)<S_2(\sigma)<\cdots<S_n(\sigma)$.

Lemma 2 *In the schedule σ for the alleged smallest counterexample I, if B_i is not a regular batch, then on each machine there is a regular batch which starts before r_i and is completed no earlier than $S_i(\sigma)$.*

Proof If B_i is not a regular batch, by the algorithm, the m machines are processing jobs during the interval $[r_i, S_i(\sigma)]$; in other words, on each machine there is a batch which starts before r_i and completes no earlier than $S_i(\sigma)$.

We use induction on i to show that the lemma holds.

Basic case: $i \le m+1$. It is obvious that B_j is a regular batch for each $1 \le j \le m$. Thus the conclusion holds.

Induction case: $i > m+1$. Assume the claim for smaller integers than i, i.e., for each $j (j < i)$; if B_j is not a regular batch, then on each machine there is a regular batch which starts before r_j and is completed no earlier than $S_j(\sigma)$.

Since B_i is not a regular batch, by Algorithm $H_m^\infty(\alpha)$, there is a batch which starts before r_i and is completed no earlier than $S_i(\sigma)$ on each machine. For contradiction, suppose one of the m batches is not a regular batch, say B_k. Then by the induction hypothesis, on each machine there is a regular batch which starts before r_k and is completed no earlier than $S_k(\sigma)$. Denote the m regular batches by $B_{k_1}, B_{k_2}, \ldots, B_{k_m}$ and assume that $S_{k_1}(\sigma) < S_{k_2(\sigma)} < \cdots < S_{k_m}(\sigma)$. Then we have that $S_{k_j}(\sigma) < S_k(\sigma) \le S_{k_j}(\sigma) + p_{k_j}$ for each $1 \le j \le m$. A job in batch k_j is released after $S_{k_{j-1}}(\sigma)$, as otherwise it would be processed in batch k_{j-1}. Therefore, $S_{k_j}(\sigma) \ge (1+\alpha) r_{k_j} + \alpha p_{k_j} > (1+\alpha) S_{k_{j-1}}(\sigma) + \alpha p_{k_j}$. Thus for each $2 \le j \le m$,

$$\begin{aligned}(1+\alpha) S_{k_j}(\sigma) &= S_{k_j}(\sigma) + \alpha S_{k_j}(\sigma) \\ &> (1+\alpha) S_{k_{j-1}}(\sigma) + \alpha p_{k_j} + \alpha S_{k_j}(\sigma) \\ &\ge (1+\alpha) S_{k_{j-1}}(\sigma) + \alpha S_k(\sigma). \end{aligned} \tag{1}$$

So,

$$\begin{aligned} S_k(\sigma) &> (1+\alpha) S_{k_m}(\sigma) + \alpha p_k \\ &> (1+\alpha) S_{k_{m-1}}(\sigma) + \alpha S_k(\sigma) + \alpha p_k \\ &> (1+\alpha) S_{k_1}(\sigma) + (m-1)\alpha S_k(\sigma) + \alpha p_k. \end{aligned} \tag{2}$$

Since B_i is not a regular batch, $S_i \le S_k + p_k$. By the algorithm, $S_i > (1+\alpha) S_k$. Therefore $p_k > \alpha S_k$. Thus,

$$\begin{aligned} S_k(\sigma) &> (1+\alpha) S_{k_1}(\sigma) + (m-1)\alpha S_k(\sigma) + \alpha p_k \\ &> (1+\alpha) S_{k_1}(\sigma) + (m-1)\alpha S_k(\sigma) + \alpha^2 S_k(\sigma). \end{aligned}$$

Therefore,

$$S_k(\sigma) > \frac{1+\alpha}{1-\alpha^2-(m-1)\alpha} S_{k_1} \ge \frac{1+\alpha}{\alpha} S_{k_1}. \tag{3}$$

Since $S_k(\sigma) \le S_{k_1} + p_{k_1}$, $S_{k_1}(\sigma) < \alpha p_{k_1}$, which contradicts that the algorithm starts the batch B_{k_1} after αp_{k_1}.

Therefore the m batches which start before r_i and are completed no earlier than $S_i(\sigma)$ are regular batches. □

Let B_l denote the first batch in σ that assumes the value $C_{\max}(\sigma)$.

Lemma 3 *In the schedule σ for the alleged smallest counterexample I, B_l is not a regular batch.*

Proof If B_l is a regular batch, then $C_{\max}(\sigma) = (1+\alpha)(r_l + p_l) \le (1+\alpha) C_{\max}(\pi)$, which contradicts our choice of I. □

Lemma 4 *In the schedule σ for the alleged smallest counterexample I, $C_{\max}(\sigma) = C_{m+1}(\sigma)$, and there are only $m+1$ batches.*

Proof Since B_l is not a regular batch, by Lemma 2, on each machine there is a regular batch which starts before r_l and is completed no earlier than $S_l(\sigma)$. By Algorithm $H_m^\infty(\alpha)$, the existence of the jobs that are completed before r_l in σ does not influence the start times of the m regular batches and B_l and the order in which the jobs are executed. Therefore, we can remove all jobs from I that are completed before the release time of B_l without changing the value $C_{\max}(\sigma)$ and without increasing the value $C_{\max}(\pi)$. Similarly, we can remove all jobs from I that are released after the start of B_l in σ. Thus there are only $m+1$ batches in σ and $C_{\max}(\sigma) = C_{m+1}(\sigma)$. □

Theorem 1 *Algorithm $H_m^\infty(\alpha)$ has a competitive ratio of $1+\alpha$, where $\alpha = (\sqrt{m^2+4} - m)/2$.*

Proof Suppose to the contrary that there exists an instance for which the algorithm finds a schedule σ with $C_{\max}(\sigma) > (1+\alpha) C_{\max}(\pi)$. Obviously, there exists a counterexample I with a minimum number of jobs. On the basis of Lemmas 1 and 4, we know that there are $m+1$ batches in σ and there is only one job in each batch of σ. For convenience, we index the $m+1$ batches in σ in the order such that $S_1(\sigma) < S_2(\sigma) < \cdots < S_{m+1}(\sigma)$. Then on the basis of Lemmas 3 and 4, we know that B_{m+1} is not a regular batch and $C_{\max}(\sigma) = C_{m+1}(\sigma)$.

Since B_{m+1} is not a regular batch, by Lemma 2, there is a regular batch that starts before r_{m+1} and is completed after S_{m+1} on each machine. It is obvious the m batches are $B_1, B_2, \ldots, B_m$. Then we will prove that $S_m > (\alpha + m - 1) p_{m+1}$.

In fact, we know that B_{m+1} is not a regular batch and for each $1 \le j \le m$,

$$S_j(\sigma) + p_j \ge S_{m+1}(\sigma). \tag{4}$$

By the algorithm, $S_1(\sigma) \ge \alpha p_1$. Thus $S_1(\sigma) + p_1 \le \frac{\alpha+1}{\alpha} S_1(\sigma)$. Therefore,

$$S_1(\sigma) \ge \frac{\alpha}{1+\alpha}(S_1(\sigma) + p_1) \ge \frac{\alpha}{1+\alpha} S_{m+1}(\sigma).$$

By the algorithm and the inequality (4), we have

$$\begin{aligned} S_j(\sigma) &> (1+\alpha) S_{j-1}(\sigma) + \alpha p_j \\ &\ge (1+\alpha) S_{j-1}(\sigma) + \alpha (S_{m+1}(\sigma) - S_j(\sigma)). \end{aligned}$$

Thus,

$$S_j(\sigma) > S_{j-1}(\sigma) + \frac{\alpha}{1+\alpha} S_{m+1}(\sigma).$$

Therefore,

$$S_m(\sigma) > \frac{m\alpha}{1+\alpha} S_{m+1}(\sigma). \tag{5}$$

Since $S_{m+1}(\sigma) > (1+\alpha)S_m(\sigma) + \alpha p_{m+1}$,

$$\begin{aligned} S_m(\sigma) &> \frac{m\alpha}{1+\alpha} S_{m+1}(\sigma) \\ &> m\alpha S_m(\sigma) + \frac{m\alpha^2}{1+\alpha} p_{m+1}. \end{aligned} \tag{6}$$

By definition of α, we have $\alpha^2 = (1 - m\alpha)$ and thus

$$S_m(\sigma) > \frac{m}{1+\alpha} p_{m+1} = (\alpha + m - 1) p_{m+1}. \tag{7}$$

Now we consider the assignment of the $m+1$ jobs in the optimal schedule π and show that the alleged counterexample is not a counterexample at all.

Case 1: There are at least two of the $m+1$ jobs are scheduled in the same batch in π. Say the two jobs are $J_i, J_j (S_i(\sigma) < S_j(\sigma))$. Then $C_{\max}(\pi) \geq r_j + \max(p_i, p_j)$. Since B_{m+1} is not a regular batch, $C_{\max}(\sigma) \leq S_i + p_i + p_{m+1}$. Thus, $C_{\max}(\sigma) - C_{\max}(\pi) \leq p_{m+1}$. Note that $C_{\max}(\pi) \geq r_{m+1} + p_{m+1} > S_m(\sigma) + p_{m+1} > (\alpha + m - 1) p_{m+1} + p_{m+1} = (\alpha + m) p_{m+1}$. Therefore,

$$\frac{C_{\max}(\sigma) - C_{\max}(\pi)}{C_{\max}(\pi)} \leq \frac{1}{\alpha + m} = \alpha,$$

i.e., $C_{\max}(\sigma)/C_{\max}(\pi) \leq 1 + \alpha$, which contradicts our choice of I.

Case 2: The $m+1$ jobs form $m+1$ batches in π.

Subcase 2.1: Any two of the m jobs $J_i (1 \leq i \leq m)$ are processed on two different machines in π. Then $C_{\max}(\pi) \geq \min_{1 \leq j \leq m}(r_j + p_j) + p_{m+1}$ and $C_{\max}(\sigma) = \min_{1 \leq j \leq m}(S_j + p_j) + p_{m+1} \leq (1+\alpha) \min_{1 \leq j \leq m}(r_j + p_j) + p_{m+1}$. Thus, $C_{\max}(\sigma)/C_{\max}(\pi) \leq 1 + \alpha$, which contradicts our choice of I.

Subcase 2.2: There are at least two of the m jobs $J_i (1 \leq i \leq m)$ which are processed on the same machine in π. Denote the two jobs by J_i and $J_j (S_i(\sigma) < S_j(\sigma))$. Then we have $C_{\max}(\pi) \geq r_i + p_i + p_j$.

If $p_j \geq p_{m+1}$, then $C_{\max}(\sigma) - C_{\max}(\pi) \leq S_i(\sigma) - r_i = \alpha(r_i + p_i)$, since $C_{\max}(\pi) \geq r_i + p_i + p_j$ and $C_{\max}(\sigma) \leq S_i(\sigma) + p_i + p_{m+1}$. It is obvious that $C_{\max}(\pi) \geq r_i + p_i$. Thus $C_{\max}(\sigma)/C_{\max}(\pi) \leq 1 + \alpha$, which contradicts our choice of I.

If $p_j \leq p_{m+1}$, then $C_{\max}(\sigma) \leq S_j(\sigma) + p_j + p_{m+1} \leq r_{m+1} + 2p_{m+1}$. Thus $C_{\max}(\sigma) - C_{\max}(\pi) \leq p_{m+1}$, since $C_{\max}(\pi) \geq r_{m+1} + p_{m+1}$. Similar to case 1, we know I is not a counterexample at all. □

3 The lower bound

Theorem 2 *Any on-line algorithm for $Pm|B = \infty, r_j$, on-line$|C_{\max}$ cannot have a competitive ratio of less than $1 + (\sqrt{m^2+4} - m)/2$.*

Proof Suppose an on-line algorithm A wants to guarantee a competitive ratio of $1+\alpha$. Since an on-line algorithm with a competitive ratio of $1 + (\sqrt{m^2+4} - m)/2$ is already given, we know that $\alpha \leq (\sqrt{m^2+4} - m)/2$. Let S_j denote the start time of J_j in the schedule produced by the on-line algorithm A. Consider the following instance provided by the adversary.

At time zero a job J_1 is released with processing time $p_1 = 1$. Since the algorithm wants to guarantee a competitive ratio of $1 + \alpha$, the algorithm A cannot start J_1 later than α, i.e., $S_1 \leq \alpha$. Then the adversary proceeds as follows. For $1 \leq i \leq m-1$, after the algorithm A starts job J_i as a batch at time S_i, the adversary proceeds by releasing a new job J_{i+1} at time $r_{i+1} = S_i + \epsilon$ with processing time $p_{i+1} = 1 - S_i$. We can show that an adversary will force the algorithm A to start J_{i+1} no later than $S_i + \alpha$, i.e., $S_{i+1} \leq S_i + \alpha$. In fact, if $S_{i+1} > S_i + \alpha$, then the makespan produced by the algorithm A will be greater than $S_i + \alpha + p_{i+1} = 1 + \alpha$; while in an optimal schedule, J_{i+1} can start at $r_{i+1} = S_i + \epsilon$ and be completed at $r_{i+1} + p_{i+1} = 1 + \epsilon$. Then, when $\varepsilon \to 0$, the algorithm A cannot guarantee a competitive ratio of $1 + \alpha$.

After the algorithm A starts J_m at $S_m \leq S_{m-1} + \alpha$, the m machines are all occupied by the m jobs until $\min_{1 \leq i \leq m}\{S_i + p_i\}$. In this situation, the adversary proceeds by releasing a new job J_{m+1} at time $r_{m+1} = S_m + \epsilon$ with processing time $p_{m+1} = \max_{1 \leq i \leq m}\{S_i + p_i - S_m\}$.

Consider the former m jobs. Let $S_0 = 0$; then we can get that $p_i = 1 - S_{i-1}$ for each $1 \leq i \leq m$ and $p_{m+1} = \max_{1 \leq i \leq m}\{S_i + p_i - S_m\} = 1 + \max_{1 \leq i \leq m}\{S_i - S_{i-1}\} - S_m$, where $S_0 = 0$.

Since $S_{i+1} \leq S_i + \alpha$ for each $1 \leq i \leq m$, $S_j \leq S_i + (j-i)\alpha$ for any $i, j (0 \leq i < j \leq m)$.

After J_{m+1} is released, it is obvious that the m machines are all occupied by the m jobs until $\min_{1 \leq i \leq m}\{S_i + p_i\}$. Therefore,

$$\begin{aligned} C_{\max} &\geq \min_{1 \leq i \leq m}\{S_i + p_i\} + p_{m+1} \\ &\geq 2 + \min_{1 \leq i \leq m}\{S_i - S_{i-1}\} + \max_{1 \leq i \leq m}\{S_i - S_{i-1}\} - S_m. \end{aligned}$$

Without loss of generality, suppose $\min_{1 \leq i \leq m}\{S_i - S_{i-1}\} = S_j - S_{j-1}$.

If $j < m$, noting that $S_{m-1} \leq S_j + (m - 1 - j)\alpha$ and $S_{j-1} \leq (j-1)\alpha$, then

$$\min_{1 \leq i \leq m}\{S_i - S_{i-1}\} + \max_{1 \leq i \leq m}\{S_i - S_{i-1}\}$$

$$\geq S_j - S_{j-1} + S_m - S_{m-1}$$
$$\geq S_m - (j-1)\alpha - (m-1-j)\alpha$$
$$\geq S_m - (m-2)\alpha.$$

If $j = m$, noting that $S_{m-1} \leq S_1 + (m-2)\alpha$, then

$$\min_{1\leq i\leq m}\{S_i - S_{i-1}\} + \max_{1\leq i\leq m}\{S_i - S_{i-1}\}$$
$$\geq S_m - S_{m-1} + S_1 - S_0$$
$$\geq S_m - (m-2)\alpha.$$

Therefore,

$$C_{\max} \geq 2 - (m-2)\alpha.$$

While in an optimal schedule, each job $J_i (1 \leq i \leq m-1)$ can form a batch which starts at r_i and J_m, J_{m+1} can form a batch which starts at r_{m+1}. Then the optimal makespan is

$$\begin{aligned} C^*_{\max} &= r_{m+1} + \max\{p_m, p_{m+1}\} \\ &= S_m + \varepsilon + 1 + \max_{1\leq i\leq m}\{S_i - S_{i-1}\} - S_m \\ &= 1 + \max_{1\leq i\leq m}\{S_i - S_{i-1}\} + \varepsilon \\ &\leq 1 + \alpha + \varepsilon. \end{aligned}$$

Since the algorithm wants to guarantee a competitive ratio of $1+\alpha$,

$$\frac{C_{\max}}{C^*_{\max}} \leq 1 + \alpha.$$

Let $\varepsilon \to 0$; we see that

$$\frac{2-(m-2)\alpha}{1+\alpha} \leq 1 + \alpha,$$
$$\alpha^2 + m\alpha - 1 \geq 0.$$

Therefore, $\alpha \geq (\sqrt{m^2+4} - m)/2$, i.e., any on-line algorithm cannot have a competitive ratio of less than $1 + (\sqrt{m^2+4} - m)/2$. □

Acknowledgements The authors would like to thank the anonymous referees whose comments very much helped to improve this paper. This research was supported by the grant 09ZR1407200 of Science Foundation of Shanghai and NSFC (10771067).

References

Brucker, P., Gladky, A., Hoogeveen, H., Kovalyvov, M. Y., Potts, C. N., Tautehahn, T., & Velde, S. L. (1998). Scheduling a batching machine. *Journal of Scheduling*, *1*, 31–54.

Chen, B., Deng, X., & Zang, W. (2001). On-line scheduling a batch processing system to minimize total weighted job completion time. In *Lecture notes in computer science* (Vol. 2223, pp. 380–389). Berlin: Springer.

Cheng, T. C. E., Ng, C. T., Yuan, J. J., & Liu, Z. H. (2004). Single machine parallel batch scheduling subject to precedence constraints. *Naval Research Logistics*, *51*, 949–958.

Graham, R. L., Lawler, E. L., Lenstra, J. K., & Rinnooy Kan, A. H. G. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals Discrete Mathematics*, *5*, 287–326.

Lee, C. Y., & Uzsoy, R. (1999). Minimizing makespan on a single batch processing machine with dynamic job arrivals. *International Journal of Production Research*, *37*, 219–236.

Lee, C. Y., Uzsoy, R., & Martin Vega, L. A. (1992). Efficient algorithms for scheduling semiconductor burn-in operations. *Operations Research*, *40*, 764–775.

Li, S., Li, G., Wang, X., & Liu, Q. (2005). Minimizing makespan on a single batching machine with release times and non-identical job sizes. *Operations Research Letters*, *33*, 157–164.

Liu, Z., & Yu, W. (2000). Scheduling one batch processor subject to job release dates. *Discrete Applied Mathematics*, *105*, 129–136.

Ng, C. T., Cheng, T. C. E., & Yuan, J. J. (2002). A note on the single machine serial batching scheduling problem to minimize maximum lateness with precedence constraints. *Operations Research Letters*, *30*, 66–68.

Ng, C. T., Cheng, T. C. E., & Yuan, J. J. (2003). The single machine batching problem with family setup times to minimize maximum lateness is strongly NP-hard. *Journal of Scheduling*, *6*, 483–490.

Nong, Q. Q., Cheng, T. C. E., & Ng, C. T. (2008). An improved online algorithm for scheduling on two unrestrictive parallel batch processing machines. *Operations Research Letters*, *36*, 584–588.

Tian, J., Fu, R., & Yuan, J. (2009). A best online algorithm for scheduling on two parallel batch machines. *Theoretical Computer Science*, *410*, 2291–2294.

Uzsoy, R. (1994). A single batch processing machine with non-identical job sizes. *International Journal of Production Research*, *32*, 1615–1635.

Yuan, J. J., Liu, Z. H., Ng, C. T., & Cheng, T. C. E. (2004). The unbounded single machine parallel batch scheduling problem with family jobs and release dates to minimize makespan. *Theoretical Computer Science*, *320*, 199–212.

Zhang, G., Cai, X., & Wong, C. K. (2001). On-line algorithms for minimizing makespan on batch processing machines. *Naval Research Logistics*, *48*, 241–258.

Zhang, G., Cai, X., & Wong, C. K. (2003). Optimal on-line algorithms for scheduling on parallel batch processing machines. *IIE Transactions*, *35*, 175–181.