依时间在线的无界平行分批机器上的最小化时间表长的研究综述

原晋江[1]

[1] 郑州大学数学系，郑州450001

摘要：在线排序是排序论中发展迅速的一个研究分支。在线排序的研究中，无界平行分批机器上的最小化时间表长模型是一个富有成果性的研究领域。本文详细综述了依时间在线的无界平行分批机器上的最小化时间表长的研究进展，并提出了若干进一步研究的问题。

关键词：运筹学、在线排序、平行批机器、时间表长、竞争比

中图分类号：O223

Online over time scheduling on unbounded p-batch machines to minimize the makespan: A survey

Jinjiang Yuan

Department of Mathematics, Zhengzhou University, Zhengzhou, Henan 450001

Abstract: Online scheduling is a rapidly developed branch in scheduling theory. In the research of online scheduling, makespan minimization on unbounded parallel-batch machines is a fruitful area. In this paper, we give a detail survey for the online over time scheduling on unbounded parallel-batch machines to minimize the makespan. Some open problems are proposed for the further research.

Key words: operations research; online scheduling; parallel-batch machines; makespan; competitive ratio.

0 Introduction

In the parallel-batch scheduling, we have n jobs $J_1, \cdots, J_n$ and m parallel-batch machines $M_1, \cdots, M_m$. Each job J_j has a release date $r_j \geq 0$ and a processing time $p_j > 0$. Each parallel-batch machine M_i can process jobs simultaneously as a batch up to b jobs. Here a batch is a subset of jobs and b is the capacity of the batches. If $b = \infty$, the model is called unbounded batching. Otherwise ($b < \infty$), the model is called bounded batching. The processing time of a batch is defined to be the maximum processing time of the jobs in the batch. A batch can be started to processing at a time t if all the jobs in the batch are released by time t. Furthermore, in a schedule π, the jobs in a batch have the same starting time and the same completion time, respectively.

Research supported by SRFDP (20070459002) and NSFC (10971201)

原晋江（1965-），男，教授，主要研究方向：组合最优化。邮箱：yuanjj@zzu.edu.cn

A schedule for parallel-batch scheduling can be determined by partitioning the jobs into batches and assigning the batches to the available time spaces of the m parallel-batch machines without overlap. In a schedule π, the completion time of a job J_j is denoted by $C_j = C_j(\pi)$. A fundamental objective function is the makespan $C_{\max} = C_{\max}(\pi)$, which is the maximum completion time of all jobs. Using the standard scheduling classification scheme of Lawler et al. [1], the makespan minimization problem on parallel-batch machines is written as

$$Pm|p\text{-}batch,\ b,\ r_j|C_{\max}.$$

Parallel-batch scheduling is motivated by semiconductor manufacturing. Uzsoy et al. [2, 3], Avramids et al. [4] and Mathirajan and Sivakumar [5] described the application in the semiconductor manufacturing process in detail. The fundamental model for bounded parallel-batch scheduling was first introduced by Lee et al. [6]. For problem $1|p\text{-}batch,\ b < \infty|C_{\max}$, it was reported by Lee and Uzsoy [7] that the optimal schedule is given by the full batch longest processing time ($FBLPT$) rule proposed first by Bartholdi. An extensive discussion of the unbounded parallel-batch scheduling problem was provided by Brucker et al. [8]. Recent developments on this topic can be found in [9] and [10]. With dynamic job arrivals and the capacity b being infinite, Lee and Uzsoy [7] presented a dynamic programming algorithm to solve problem $1|p\text{-}batch,\ b = \infty,\ r_j|C_{\max}$ in $O(n^2)$ time. For the same problem, Poon and Zhang [11] presented an improved $O(n \log n)$-time algorithm. For the bounded parallel-batch scheduling problem $1|p\text{-}batch,\ b < \infty,\ r_j|C_{\max}$, Liu and Yu [12] showed that the problem with only two arrival times is NP-hard, and gave a pseudo-polynomial-time algorithm in case of fixed number of arrival times. Brucker et al. [8] proved that the general problem is NP-hard in the strong sense.

Online scheduling is a relatively new topic of scheduling research and has been extensively studied in the last decade. While there are different meanings of online scheduling, the term "online" in this paper means that jobs arrive over time.

In the online over time environment, jobs arrive over time and we do not have any information about the jobs in advance. The information of each job J_j can be known only at the arrival time r_j of the job. Hence, we must schedule the available jobs without the information of the the future jobs. Usually, the information of a job J_j includes its release time r_j, its processing time p_j, its delivery time d_j, its weight w_j, and its job family, etc. In the makespan minimization, the necessary information of a job is its release time r_j, its processing time p_j, and its job family (when the jobs are partitioned into no compatible job families).

The quality of an online algorithm is measured by its competitive ratio. Suppose that we are considering an online scheduling problem $\mathcal{P}$ to minimize a certain objective function. Let $C_{\text{on}}(L)$ and $C_{\text{opt}}(L)$ denote, respectively, the objective value of an online algorithm H and of an optimal off-line algorithm for an input job list L. The competitive ratio R_H of algorithm H is defined as

$$R_H = \sup_L \{C_{\text{on}}(L)/C_{\text{opt}}(L)\}.$$

In this case, we also say that algorithm H is R_H-competitive and R_H is an upper bound of the

competitive ratio of problem $\mathcal{P}$

Given an online scheduling problem $\mathcal{P}$, we say that the online algorithms for $\mathcal{P}$ has a lower bound ρ, if every online algorithm for $\mathcal{P}$ has a competitive ratio at least ρ. Furthermore, if $\mathcal{A}$ is an online algorithm for $\mathcal{P}$ such that the competitive ratio of $\mathcal{A}$ is exactly the lower bound ρ for $\mathcal{P}$, we say that $\mathcal{A}$ is a best possible online algorithm for $\mathcal{P}$.

Some examples of studies on online scheduling problems (with jobs arriving over time) are [13], [14], [15], [16] and [17], among others.

In general, the competitive ratio of an online algorithm will improve if some information on the jobs is known in advance. This scenario is described as "semi-online" in the literature. In the literature there are plenty researches concerning semi-online scheduling with jobs arriving over a list. For example, Cheng et al. [18] studied the semi-online scheduling on parallel machines with given total processing time. Seiden et al. [19] studied the semi-online scheduling on parallel machines with decreasing job sizes. Tan and He [20] studied semi-online scheduling on two parallel machines with combined partial information. In contrast, there are only a few researches concerning semi-online scheduling with jobs arriving over time. The representative publication is given by Hall et al. [21]. They studied the semi-online scheduling on a single machine to minimize the sum of weighted completion time with known arrival times of the jobs.

In this survey, we report the developments on the online scheduling on the unbounded parallel-batch machines to minimize the makespan. The scheduling models include the following forms.

$$1|\text{online, p-batch, } b=\infty|C_{\max},$$

$$1|\text{online, p-batch, restart, } b=\infty|C_{\max},$$

$$1|\text{online, p-batch, L-restart, } b=\infty|C_{\max},$$

$$1|\text{online, p-batch, } J^*(t),\ b=\infty|C_{\max},$$

$$1|\text{online, p-batch, } p^*(t),\ b=\infty|C_{\max},$$

$$1|\text{online, p-batch, } r^*(t),\ b=\infty|C_{\max},$$

$$1|\text{online, p-batch, families, } b=\infty|C_{\max},$$

$$Pm|\text{online, p-batch, } b=\infty|C_{\max},$$

$$Pm|\text{online, p-batch, families, } b=\infty|C_{\max}.$$

The notations appearing in the above models are described as follows.

• "restart" means that a running task may be interrupted, losing all the work done on it. The jobs in the interrupted task are then released and become independently unscheduled jobs. Allowing restarts reduces the impact of a wrong decision.

• "L-restart" means that batches are only allowed *limited restarts*. If a batch has been restarted one time, then all jobs in it are considered as interrupted jobs. Any new batch that contains interrupted jobs cannot be restarted any more. That is, any job is allowed to restart only once.

• "$J^*(t)$" means that at time t the information of the first (equivalently, the last) longest job arriving after time t is given.

• "$p^*(t)$" means that at time t the processing time of the first (equivalently, the last) longest job arriving after time t is given.

• "$r^*(t)$" means that at time t the arrival time of the first (equivalently, the last) longest job arriving after time t is given.

• "families" means that the jobs are partitioned into incompatible families so that the jobs in different families cannot be processed in a common batch.

In the research of online scheduling on parallel-batch machines, the following notations are widely accepted.

• $U(t)$ is the set of available jobs at time t.
• $p(t)$ is the processing time of $U(t)$.
• $J(t)$ is one of the last longest jobs in $U(t)$.
• $r(t)$ is the arrival time of $J(t)$.

1 Problem $1|$online, p-batch, $b = \infty|C_{\max}$

The known lower bound and the upper bound of the problem is given in the following table.

lower bound	upper bound	situation
1.618	1.618	best possible

Zhang, Cai and Wong [17] and Deng, Poon and Zhang [14] independently provided the following best possible online algorithm.

Algorithm 1 At time t, if $U(t) \neq \emptyset$ and $t \geq (1+\alpha)r(t) + \alpha p(t)$, then start to processing $U(t)$ as a single batch at time t. Otherwise, do nothing but wait.

Poon and Yu [22] provided the following flexible online algorithm, which is also best possible.

Algorithm 2 At time t, if $U(t) \neq \emptyset$ and $t \geq \lambda(t)$, where $\lambda(t)$ is an arbitrary value satisfying $\alpha p(t) \leq \lambda(t) \leq (1+\alpha)r(t) + \alpha p(t)$, then start to processing $U(t)$ as a single batch at time t. Otherwise, do nothing but wait.

The above two algorithms provided basic ideas for the research of online scheduling on parallel-batch machines, which have been widely accepted in the onward research.

2 Problem $1|$online, p-batch, restart, $b=\infty|C_{\max}$

The known lower bound and the upper bound of the problem is given in the following table.

lower bound	upper bound	situation
1.382	1.382	best possible

The lower bound was given by Fu, Tian, Yuan and Lin [23].

The following algorithm was given by Yuan, Fu, Ng and Cheng [24].

The following notations are used in the algorithm.

• For a batch B_k, J_k is the last longest job in B_k. We call J_k the key job of B_k.

• $\alpha=(3-\sqrt{5})/2\approx 0.382$. Note that $\alpha^2-3\alpha+1=0$.

• $x=1-\alpha\approx 0.618$. Note that $x^2+x-1=0$ and $\alpha=x/(1+x)=1/(2+x)$.

• For $i\geq 0$, $f(i)=\sum_{0\leq j\leq i-1}x^j$. Note that $f(0)=0$ and $f(i+1)=f(i)+x^i$.

• For a job J_k and a time instant $t\geq r_k$, $i(t,k)=\max\{i: f(i)p_k\leq t-r_k\}$. When no confusion may arise, we write $i(t)$ for $i(t,k)$ for brevity. Note that $t-r_k$ is the length of the time period measured from the arrival of job J_k. So $i(t)$ increases with t.

Algorithm H

Step 0: Set $t=0$.

Step 1: At time t, if $U(t)=\emptyset$, go to Step 4. Otherwise schedule all the jobs in $U(t)$ as a single batch $B_k=U(t)$ starting at time t. Find the key job J_k of B_k. Calculate $i(t)=i(t,k)$.

Step 2: In time interval $(t,t+p_k)$, if no new job arrives, set $t=t+p_k$ and go to Step 1.

Step 3: If a new job J_h arrives at time $r<t+p_k$, do the following:

(3.1) If $p_h\geq p_k$, interrupt the running batch B_k and restart a new batch at time r: reset $t=r$ and go to Step 1.

(3.2) If $p_h<p_k$ and $r-r_k-f(i(t))p_k<x^{i(t)+1}p_k$, do the following:

(3.2.1) If $p_h\geq r-r_k-f(i(t))p_k$, interrupt the running batch B_k and restart a new batch at time r: reset $t=r$ and go to Step 1.

(3.2.2) If $p_h<r-r_k-f(i(t))p_k$, continue processing the present batch B_k and then go to Step 2.

(3.3) If $p_h<p_k$ and $r-r_k-f(i(t))p_k\geq x^{i(t)}p_k$, continue processing the present batch B_k and then go to Step 2.

(3.4) If $p_h<p_k$ and $x^{i(t)+1}p_k\leq r-r_k-f(i(t))p_k<x^{i(t)}p_k$, do the following:

(3.4.1) If $p_h<x^{i(t)+1}p_k$, continue processing the present batch B_k and then go to Step 2.

(3.4.2) If $p_h\geq x^{i(t)+1}p_k$, interrupt the running batch B_k at time r, and do the following (delay):

(3.4.2.1) If there is a new job J with processing time $p\geq p_k$ arriving at time $t^*\in(r,r_k+f(i(t)+1)p_k)$, restart a new batch at time t^*: reset $t=t^*$ and go to Step 1.

(3.4.2.2) Otherwise restart a new batch at time $r_k + f(i(t)+1)p_k$: reset $t = r_k + f(i(t)+1)p_k$ and go to Step 1.

Step 4: If there still are some jobs arriving, set t as the arrival time of the first job and go to Step 1; otherwise stop and complete the schedule at time t.

3 Problem $1|$online, p-batch, L-restart, $b = \infty|C_{\max}$

The known lower bound and the upper bound of the problem is given in the following table.

lower bound	upper bound	situation
1.5	1.5	best possible

Fu, Tian and Yuan [25] presented the following best possible algorithm.

Algorithm H

Step 0: Set $t = 0$, $s = 0$, $\delta = 0$.

Step 1: At time t, if $U(t) = \emptyset$, go to Step 5. Otherwise, find the last longest job J_k in $U(t)$.

Step 2: If $t \geq \frac{1}{2}p_k$, then set $s = t$ and schedule all jobs in $U(s)$ as a single batch immediately. Otherwise, $t < \frac{1}{2}p_k$. Then reset $t = \frac{1}{2}p_k$ and go to Step 1.

Step 3: In the time interval $(s,\ s+p_k)$, if no new jobs arrive or $\delta = 1$, set $t = s + p_k$ and $\delta = 0$, go to Step 1.

Step 4: If $\delta = 0$ and a new job J_h arrives at time $r_h < s + p_k$, do the following:

4.1: If $r_h \geq \frac{5}{4}p_k$, set $t = s + p_k$ and go to Step 1.

4.2: If $p_h \leq \frac{3}{4}p_k$, go on processing the present batch B_k and go to Step 3.

4.3: If $p_h \geq \frac{3}{2}p_k$, set $t = s + p_k$ and go to Step 1.

4.4: If $p_k \leq p_h < \frac{3}{2}p_k$, then do the following:

4.4.1: If $r_h \geq p_h$, restart the running batch B_k at time moment r_h: set $t = r_h$ and $\delta = 1$, go to Step 1.

4.4.2: If $r_h < p_h$, go on processing B_k in the time interval $[r_h, p_h]$ unless a new job J^* with $p^* \geq p_h$ arrives.

If there is a new job J^* with $p^* \geq p_h$ arriving no later than time moment p_h, set $J_h = J^*$ and go to Step 4.1.

Otherwise, no new job J^* with $p^* \geq p_h$ arrives in the time interval $(r_h, p_h]$. Restart the running batch B_k at time p_h: set $t = p_h$ and $\delta = 1$, go to Step 1.

4.5: If $\frac{3}{4}p_k < p_h < p_k$, go on processing B_k in the time interval $[r_h, \frac{5}{4}p_k]$ unless a new job J^* with $p^* \geq p_h$ arrives.

If there is a new job J^* with $p^* \geq p_h$ arriving no later than time moment $\frac{5}{4}p_k$, set $J_h = J^*$ and go to Step 4.1.

Otherwise, no new job J^* with $p^* \geq p_h$ arrives in the time interval $[r_h, \frac{5}{4}p_k]$. Restart the running batch B_k at time $\frac{5}{4}p_k$: set $t = \frac{5}{4}p_k$ and $\delta = 1$, go to Step 3.

Step 5: If there still are some jobs arriving, set t as the arrival time of the first job and go to Step 1; otherwise stop and complete the schedule at time t.

4 Problem $1|$online, p-batch, $J^*(t)$ $(p^*(t))$, $b = \infty|C_{\max}$

The known lower bound and the upper bound of the problem is given in the following table.

lower bound	upper bound	situation
1.382	1.382	best possible

We in fact have two problems:

- $1|$online, p-batch, $J^*(t)$, $b = \infty|C_{\max}$.
- $1|$online, p-batch, $p^*(t)$, $b = \infty|C_{\max}$.

Yuan, Ng and Cheng [26] presented the following best possible algorithm for the both problems.

The following notations are used.

- r_0 is the release date of the first longest job.
- $U(t)$ is the set of unprocessed jobs available at time instant t.
- $J(t)$ is the first longest job in $U(t)$. The arrival time and processing time of $J(t)$ are denoted by $r(t)$ and $p(t)$, respectively.
- $J^*(t)$ is the first longest job arriving after time instant t. The arrival time and processing time of $J^*(t)$ are denoted by $r^*(t)$ and $p^*(t)$, respectively. If no job arriving after t, we set $p^*(t) = 0$.
- $p_{\max}$ is the maximum processing time of all the jobs. Then $p_{\max} = p(r_0)$.
- $\alpha = (3 - \sqrt{5})/2 \approx 0.382$. Note that $\alpha^2 - 3\alpha + 1 = 0$.
- $x = 1 - \alpha \approx 0.618$. Note that $x^2 + x - 1 = 0$ and $\alpha = x/(1 + x) = 1/(2 + x)$.
- For $i \geq 0$, $f(i) = \sum_{0 \leq j \leq i-1} x^j$. Note that $f(0) = 0$.
- $f(\infty) = \sum_{j=0}^{\infty} x^j = 1/(1 - x) = 1/\alpha$.
- For a time instant t with $p(t) > 0$, $i(t) = \max\{i : f(i)p(t) \leq t\}$. We define $i(t) = \infty$ if $t \geq f(\infty)p(t)$.

Note that when $t < f(\infty)p(t) = p(t)/\alpha$ (equivalently, $i(t) < \infty$), we have $f(i(t))p(t) \leq t < f(i(t) + 1)p(t)$.

Algorithm H

Step 0: Set $t := r_0$, $D := \{r_0\}$ and $\mathcal{J} := \{J(r_0)\}$.

Step 1: If $U(t) = \emptyset$, do the following.

(1.1) If $p^*(t) = 0$, terminate the algorithm.

(1.2) If $p^*(t) > 0$, reset $D := D \cup \{r^*(t)\}$, $\mathcal{J} := \mathcal{J} \cup \{J^*(t)\}$ and $t := r^*(t)$.

Step 2: If $i(t) = \infty$, then go to Step 6.

Step 3: If $t \geq f(i(t))p(t) + x^{i(t)+1}p(t)$, reset $D := D \cup \{f(i(t)+1)p(t)\}$ and $t := f(i(t)+1)p(t)$.

Step 4: If $p^*(t) = 0$, then schedule $U(t)$ as a single batch starting at time t and terminate the algorithm.

Step 5: Do the following.

(5.1) If $t - f(i(t))p(t) < p^*(t) < x^{i(t)+1}p(t)$, then wait for the first time instant $t^* \in (t, f(i(t))p(t) + p^*(t)]$ such that either $t^* = f(i(t))p(t) + p^*(t)$ or job $J^*(t)$ arrives at time t^*. Reset $D := D \cup \{t^*\}$. In the later case, reset $\mathcal{J} := \mathcal{J} \cup \{J^*(t)\}$. Reset $t := t^*$ and go back to Step 4.

(5.2) If $t - f(i(t))p(t) < x^{i(t)+1}p(t) \leq p^*(t)$, then reset $D := D \cup \{r^*(t)\}$, $\mathcal{J} := \mathcal{J} \cup \{J^*(t)\}$ and $t := r^*(t)$. Return to Step 2.

(5.3) If $p^*(t) \leq t - f(i(t))p(t) < x^{i(t)+1}p(t)$, then schedule $U(t)$ as a single batch starting at time t. Reset $D := D \cup \{t + p(t)\}$, $t := t + p(t)$, and go to Step 6. (Note that, after updating, if $U(t) \neq \emptyset$, we have $i(t) = \infty$.)

Step 6: Reset t as the first time instant $t^* \geq t$ with $p^*(t^*) = 0$. Reset $D := D \cup \{t^*\}$. Then schedule $U(t)$ as a single batch starting at time t and terminate the algorithm. □

5 Problem $1|$online, p-batch, $r^*(t)$, $b = \infty|C_{\max}$

The known lower bound and the upper bound of the problem is given in the following table.

lower bound	upper bound	situation
1.442	1.5	unclosed

Yuan, Ng and Cheng [26] presented the following algorithm.

Algorithm H_r

• $\alpha = 3/2$.

Step 0: Set $t := 0$, and $D := \{0\}$. Define $r_0 = 0$. Here we assume that the first longest job arrives at time 0.

Step 2: If $r^*(t) = \infty$, then schedule $U(t)$ as the first batch starting at time t and terminate the algorithm.

Step 3: If $r^*(t) < t + \alpha p_0$ and $p_t > \alpha(r^*(t) + p_0)$, then wait for time instant $r^*(t)$. Reset $t := r^*(t)$. Reset $D := D \cup \{t\}$ and back to Step 2.

Step 4: If either $r^*(t) \geq t + \alpha p_0$ or $p_t \leq \alpha(r^*(t) + p_0)$, then schedule $U(t)$ as the first batch starting at time t. Define $t_0 = t$ and $r_1 = r^*(t)$. Reset $t := \max\{r^*(t), t + p_0\}$. Reset $D := D \cup \{t\}$.

Step 5: Schedule $U(t)$ as the second batch starting at time t. If $r^*(t) = \infty$, then terminate the algorithm. Otherwise, reset $D := D \cup \{t + p_{r_1}\}$ and $t := t + p_{r_1}$.

Step 6: Wait for the latest time instant $t^* \geq t$ such that $r^*(t^*) = \infty$, schedule $U(t^*)$ as the third batch starting at time t^*, and terminate the algorithm. □

6 Problem $1|$online, p-batch, families, $b = \infty|C_{\max}$

The known lower bound and the upper bound of the problem is given in the following table.

m	lower bound	upper bound	situation
$f \to \infty$	2	2	best possible
$f = 2$	1.7808	1.7808	best possible
f arbitrary	$1 + \frac{\sqrt{4f^2-1}}{2f}$	$1 + \frac{\sqrt{4f^2-1}}{2f}$	best possible

- The first result was obtained by Nong, Yuan, Fu, Lin and Tian [27].
- The second result was obtained by Fu, Tian and Yuan [28].
- The third result was obtained by Fu, Cheng, Ng and Yuan [29].

Write $\alpha_f = \frac{\sqrt{4f^2-1}}{2f}$. The following is the algorithm for arbitrary f.

Algorithm $A(\alpha_f)$

At time t, if there exist some waiting batches, say, $B_1, \cdots, B_i$ with $p_1 \geq p_2 \geq \cdots \geq p_i$, then start to processing B_1 as a single batch if $t \geq \alpha_f(p_1 + p_2 + \cdots + p_i)$; otherwise, do nothing but wait.

7 Problem $Pm|$online, p-batch, $p_j = 1$, $b = \infty|C_{\max}$

Let β_m be the positive solution of equation $(1+\beta_m)^{m+1} = \beta_m + 2$. The known lower bound and the upper bound of the problem is given in the following table.

lower bound	upper bound	situation
$1 + \beta_m$,	$1 + \beta_m$	best possible

Zhang, Cai and Wong [30] presented the following best possible algorithm.

Algorithm $A^{\infty}(\beta_m)$**:** At time t, if a machine is idle, $U(t) \neq \emptyset$, and $t \geq (1+\beta_m)r(t)+\beta_m$, then start $U(t)$ as a single batch on the machine at time t. Otherwise, do nothing but wait.

8 Problem $Pm|$online, p-batch, $b = \infty|C_{\max}$

The known lower bound and the upper bound of the problem is given in the following table.

lower bound	upper bound	situation
$1+(\sqrt{m^2+4}-m)/2$,	$1+(\sqrt{m^2+4}-m)/2$	best possible

Liu, Lu and Fang [31] and Tian, Cheng, Ng and Yuan [32] independently presented two distinct best possible algorithms. Their result generalized the work of Nong, Cheng and Ng [33] and Tian, Fu and Yuan [34] for the case of $m = 2$.

The algorithm of Liu, Lu and Fang [31] can be stated as follows.

Algorithm $H_m^{\infty}(\alpha)$: At time t, if a machine is idle, $U(t) \neq \emptyset$, and $t \geq (1+\alpha)r(t)+\alpha p(t)$, then start $U(t)$ as a single batch on the machine at time t. Otherwise, do nothing but wait.

The algorithm of Tian, Cheng, Ng and Yuan [32] can be stated as follows.

Algorithm $H(\alpha_m)$: At time t, if a machine is idle, $U(t) \neq \emptyset$, and $t \geq \eta(t) = S^*(t) + \alpha_m p_{\max}(t)$, then start $U(t)$ as a single batch on the machine at time t; otherwise, do nothing but wait.

9 Problem $P2|$online, p-batch, 2 families, $b = \infty|C_{\max}$

The known lower bound and the upper bound of the problem is given in the following table.

lower bound	upper bound	situation
1.618	1.618	best possible

Fu, Cheng, Ng and Yuan [35] presented the following best possible algorithm.

Algorithm H_2**:** At time t, if at least one machine is idle, $U_1(t) \neq \emptyset$ and $U_2(t) = \emptyset$, then start $U_1(t)$ on an idle machine if and only if $t \geq \alpha p_1(t)$. If both machines are idle, $U_1(t) \neq \emptyset$ and $U_2(t) \neq \emptyset$, start $U_1(t)$ and $U_2(t)$ on different machines if and only if $t \geq \alpha p_1(t)$. If only one machine is idle, $U_1(t) \neq \emptyset$ and $U_2(t) \neq \emptyset$, start $U_1(t)$ as a single batch on the machine if and

only if $t \geq \alpha p_1(t) + p_2(t)$. Otherwise, do nothing but wait.

10 Problem $Pm|$online, p-batch, m families, $b = \infty|C_{\max}$

The known lower bound and the upper bound of the problem is given in the following table.

lower bound	upper bound	situation
1.618	1.707	uncloosed

The corresponding research was presented by Tian, Cheng, Ng and Yuan [36].

11 $P2|$online, p-batch, L-restart, $b = \infty|C_{\max}$

lower bound	upper bound	situation
1.298	1.366	uncloosed

The corresponding result was presented by Fu, Cheng, Ng and Yuan [37].
Let $\alpha = \frac{\sqrt{3}-1}{2}$. Then $2\alpha(1+\alpha) = 1$. The online algorithm is described as follows.

Algorithm AR

Step 0: Set t to be the first arrival time of the jobs and consider both machines at time t.

Step 1: If both machines are idle, do the following: If $t < \alpha p_t$, set t as the minimum value between αp_t and the next arrival time, then update p_t and go back to Step 1; if $t \geq \alpha p_t$, start processing B_t at once on any idle machine and go to Step 5.

Step 2: If one machine is idle and the other machine is processing some batch, say B_a, do the following: If $t < \max\{S_a + \alpha p_a, \alpha p_t\}$, set $t = \max\{S_a + \alpha p_a, \alpha p_t\}$, update p_t, and go to Step 1; if $t \geq \max\{S_a + \alpha p_a, \alpha p_t\}$, start processing B_t at once on the idle machine and go to Step 5.

Step 3: If there are two running batches, say B_a and B_b with $S_a < S_b$, do the following:

Step 3.1: If B_b is a restart-batch, go to Step 6.

Step 3.2: If either $\frac{p_b}{p_a} \geq (1+\alpha)$ or $\frac{p_b}{p_a} \leq \frac{1}{2+\alpha}$, go to Step 6.

Step 3.3: If $\frac{p_b}{p_a} \in (1, 1+\alpha)$, do the following:

(3.3.1) If $t < S_a + \frac{1}{2}p_a$, set $t = S_a + \frac{1}{2}p_a$ and go to Step 3.3.2;

(3.3.2) If $t = S_a + \frac{1}{2}p_a$ and $p_t \in (\frac{1}{2}p_a, p_a)$, go to Step 4;

(3.3.3) Otherwise, go to Step 6.

Step 3.4: If $\frac{p_b}{p_a} \in (1-\alpha, 1]$, do the following:

(3.4.1) If $t < S_a + \frac{1}{2}p_a$, set $t = S_a + \frac{1}{2}p_a$ and go to Step 3.4.2;

(3.4.2) If $t = S_a + \frac{1}{2}p_a$ and $p_t \in (\frac{1}{2}p_a, \frac{1}{1-\alpha}p_b)$, go to Step 4;

(3.4.3) Otherwise, go to Step 6.

Step 3.5: If $\frac{p_b}{p_a} \in (\frac{1}{2+\alpha}, 1-\alpha]$, do the following:

(3.5.1) If $t < \max\{S_a + \frac{1}{2}p_a, p_a - \alpha^2 p_b\}$, set $t = \max\{S_a + \frac{1}{2}p_a, p_a - \alpha^2 p_b\}$ and go to Step 3.5.2;

(3.5.2) If $t = \max\{S_a + \frac{1}{2}p_a, p_a - \alpha^2 p_b\}$ and $p_t \in (2\alpha^2 p_a + \alpha p_b, \frac{1}{1-\alpha}p_b)$, go to Step 4;

(3.5.3) Otherwise, go to Step 6.

Step 4: Restart the running batch B_b and start processing all the waiting jobs at once.

Step 5: Reset t as the arrival time of the next job and go to Step 1.

Step 6: Reset t as the next completion time and go to Step 1.

12 Open Problems

We present the following open problems for the further research.

(1) $Pm|$online, p-batch, f families, $b = \infty|C_{\max}$.

(2) $Pm|$online, p-batch, restart, $b = \infty|C_{\max}$.

(3) $Pm|$online, p-batch, L-restart, $b = \infty|C_{\max}$.

(4) $Pm|$online, p-batch, $J^*(t)$, $b = \infty|C_{\max}$.

(5) $Pm|$online, p-batch, $p^*(t)$, $b = \infty|C_{\max}$.

(6) $Pm|$online, p-batch, $r^*(t)$, $b = \infty|C_{\max}$.

Furthermore, for problem 1|online, p-batch, $b < \infty|C_{\max}$, the known lower bound is 1.618 and the known upper bound is 2. Poon and Yu [38] presented an online algorithm of competitive ratio 7/4 when $b = 2$. Foe general b, it remains a long standing and challenging open problem.

参考文献

[1] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan and D.B. Shmoys, Sequencing and scheduling: Algorithms and complexity, in Logistics of Production and Inventory; Handbooks Operation Research Management Science 4, S.C. Graves, P.H. Zipkin, and A.H.G. Rinnooy Kan, eds., North-Holland, Amsterdam, (1993), 445-522.

[2] R. Uzsoy, C.Y. Lee and L.A. Martin-Vega, A review of production planning and scheduling models in the semiconductor industry, part I: System characteristics, performance evaluation and production planning, IIE Transaction on Scheduling and Logistics, 24(1992), 47-61.

[3] R. Uzsoy, C.Y. Lee and L.A. Martin-Vega, A survey of production planning and scheduling models in the semiconductor industry, part II: Shop-floor control, IIE Transaction on Scheduling and Logistics, 26(1994), 44-55.

[4] A.N. Avramidis, K.J. Healy and R. Uzsoy, Control of a batch processing machine: a computational approach, International Journal of Production Research, 36(1998), 3167-3181.

[5] M. Mathirajan and A.I. Sivakumar, A literature review, classification and simple meta-analysis on scheduling of batch processors in semiconductor, The International Journal of Advanced Manufacturing Technology, 29(2006), 990-1001.

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

[7] C.Y. Lee and R. Uzsoy, Minimizing makespan on a single batch processing machine with dynamic job arrivals, International Journal of Production Research, 37(1999), 219-236.

[8] P. Brucker, A. Gladky, H. Hoogeveen, M.Y. Kovalyov, C.N. Potts, T. Tautenhahn and S.L. van de Velde, Scheduling a batching machine, Journal of Scheduling, 1(1998), 31-54.

[9] P. Brucker, Scheduling Algorithms, Springer, Berlin, 2003.

[10] P. Brucker and S. Knust, Complexity Results for Scheduling Problems, http://www.mathematik.uniosnabrueck.de/research/OR/class/, 2007.

[11] C.K. Poon, P.X. Zhang, Minimizing makespan in batch machine scheduling, Algorithmica, 39 (2004) 155-174.

[12] Z. Liu and W. Yu, Scheduling one batch processor subject to job release date, Discrete Applied Mathematics, 105(2000), 129-136.

[13] E.J. Anderson and C.N. Potts, Online scheduling of a single machine to minimize total weighted completion time, Mathematics of Operations Research, 29(2004), 686-697.

[14] X.T. Deng, C.K. Poon and Y.Z. Zhang, Approximation algorithms in batch processing, Journal of Combinatorial Optimization, 7(2003), 247-257.

[15] J.A. Hoogeveen and A.P.A. Vestjens, Optimal on-line algorithms for single-machine scheduling, Lecture Notes in Computer Science, 1084(1996), 404-414.

[16] J.A. Hoogeveen and A.P.A. Vestjens, A best possible deterministic on-line algorithm for minimizing maximum delivery time on a single machine, SIAM Journal on Discrete Mathematics, 13(2000), 56-63.

[17] G.C. Zhang, X.Q. Cai and C.K. Wong, Online algorithms for minimizing makespan on batch processing machines, Naval Research Logistics, 48(2001), 241-258.

[18] T.C.E. Cheng, H. Kellerer and V. Kotov, Semi-on-line multiprocessor scheduling with given total processing time, Theoretical Computer Science, 337(2005), 134-146.

[19] S. Seiden, J. Sgall and G. Woeginger, Semi-online scheduling with decreasing job sizes, Operations Research Letters, 27(2000), 215-221.

[20] Z.Y. Tan, Y. He, Semi-on-line problems on two identical machines with combined partial information, Operations Research Letters, 30(2002), 408-414.

[21] N.G. Hall, M.E. Posner and C.N. Potts, Online scheduling with known arrival times, Mathematics of Operations Research, 34(2009), 92-102.

[22] C.K. Poon and W.C. Yu, A flexible online scheduling algorithms for batch machine with infinite capacity, Annals of Operations Research, 133(2005), 175-181.

[23] R.Y. Fu, J. Tian, J.J. Yuan and Y.X. Lin, On-line scheduling in a parallel batch processing system to minimize makespan using restarts, Theoretical Computer Science, 374(2007), 196-202.

[24] J.J. Yuan, R.Y. Fu, C.T. Ng and T.C.E. Cheng, A best on-line algorithm for unbounded parallel batch scheduling to minimize makespan with restarts, Journal of Scheduling, DOI: 10.1007/s10951-010-0172-2.

[25] R.Y. Fu, J. Tian, J.J. Yuan and C. He, On-line scheduling on a batch machine to minimize makespan with limited restarts, Operations Research Letters, 36(2008), 255-258.

[26] J.J. Yuan, C.T. Ng and T.C.E. Cheng, Best semi-online algorithms for unbounded parallel batch scheduling, Discrete Applied Mathematics, Accepted for Publication.

[27] Q.Q. Nong, J.J. Yuan, R.Y. Fu, L. Lin and J. Tian, The single-machine parallel-batching on-line scheduling problem with family jobs to minimize makespan, Internal Journal of Production Economics, 111(2008), 435-440.

[28] R.Y. Fu, J. Tian and J.J. Yuan, On-line scheduling on an unbounded batch machine to minimize makespan of two families of jobs, Journal of Scheduling, 12(2009), 91-97.

[29] R.Y. Fu, T.C.E. Cheng, C.T. Ng and J.J. Yuan, A best online algorithm for a single parallel batch machine scheduling with f job families, (in submission).

[30] G.C. Zhang, X.Q. Cai and C.K. Wong, Optimal online algorithms for scheduling on parallel batch processing machines, IIE Transactions, 35(2003), 175-181.

[31] P.H. Liu, X.W. Lu and Y. Fang, A best possible deterministic on-line algorithm for minimizing makespan on parallel batch machines. Journal of scheduling, DOI 10.1007/s10951-009-0154-4.

[32] J. Tian, T.C.E. Cheng, C.T. Ng and J.J. Yuan, Online scheduling on unbounded parallel-batch machines to minimize makespan, Information Processing Letters, 109(2009), 1211-1215.

[33] Q.Q. Nong, T.C.E. Cheng and C.T. Ng, An improved on-line algorithm for scheduling on two unrestrictive parallel batch processing machines, Operations Research Letters, 36(2008), 584-588.

[34] J. Tian, R.Y. Fu and J.J. Yuan, A best online algorithm for scheduling on two parallel batch machines, Theoretical Computer Science, 410(2009), 2291-2294.

[35] R.Y. Fu, T.C.E. Cheng, C.T. Ng and J.J. Yuan, A best online algorithm for scheduling on two parallel-batch machines with infinite batch size to minimize makespan, (in submission).

[36] J. Tian, T.C.E. Cheng, C.T. Ng and J.J. Yuan, Online scheduling on unbounded parallel-batch machines with incompatible job families, (in submission).

[37] R.Y. Fu, T.C.E. Cheng, C.T. Ng and J.J. Yuan, Online scheduling on two parallel-batching machines with limited restarts to minimize the makespan, Information Processing Letters, 110(2010), 444-450.

[38] C.K. Poon and W.C. Yu, On-line scheduling algorithms for a batch machine with finite capacity, Journal of Combinatorial Optimization, 9(2005), 167-186.