J Comb Optim (2011) 22:509–516
DOI 10.1007/s10878-010-9298-6

Optimal on-line algorithms for one batch machine with grouped processing times

Yang Fang · Peihai Liu · Xiwen Lu

Published online: 12 February 2010

Abstract In this paper, we study on-line scheduling problems on a batch machine with the assumption that all jobs have their processing times in $[p, (1+\phi)p]$, where $p>0$ and $\phi = (\sqrt{5}-1)/2$. Jobs arrive over time. First, we deal with the on-line problem on a bounded batch machine with the objective to minimize makespan. A class of algorithms with competitive ratio $(\sqrt{5}+1)/2$ are given. Then we consider the scheduling on an unbounded batch machine to minimize the time by which all jobs have been delivered, and provide a class of on-line algorithms with competitive ratio $(\sqrt{5}+1)/2$. The two class of algorithms are optimal for the problems studied here.

Keywords Batching · Scheduling · On-line algorithm · Delivery time

1 Introduction

A batch machine is a machine that can process up to B jobs simultaneously as a batch. All jobs processed in one batch have the same starting time and completion time. The batch model studied in this paper is p-batch model (burn-in model). This model is motivated by the burn-in operations in the final testing stage of semiconductor manufacturing, and one job's processing time is its minimal burn-in time. The processing time of one batch is equal to the longest processing time of the jobs in it (Lee et al. 1992). Once one batch starts to be processed, we can not stop it.

There are unbounded and bounded machines. For unbounded machine, there is no upper bound on the number of jobs that can be processed in the same batch, i.e., $B=+\infty$. On bounded machine each batch size B is limited, i.e., $B<+\infty$. Note

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

that in classical scheduling problems the machine can process no more than one job at a time. So it can be regarded as the special case $B = 1$ of bounded batch machine.

Scheduling problems are divided into two categories: off-line and on-line. Most works in scheduling area are for the off-line problem, i.e., before we decide to schedule, we know all information about every job J_j. However, in reality it is unreasonable to assume that all knowledge of the jobs are available beforehand. In this paper, we study the on-line model where jobs arrive over time. Every job has one release date r_j, before which we do not have any information about the job. As one job arrives, we get all of its information, such as processing time p_j and delivery time q_j. We need to decide whether to start the jobs or wait for more information.

For a given on-line scheduling problem, we usually use the competitive ratio to measure the performance of an on-line algorithm. Let $A(I)$ and $OPT(I)$ denote, respectively, the values obtained by an on-line algorithm A and an optimal off-line algorithm for an input instance I. The algorithm is ρ-competitive, if $A(I) \le \rho \cdot OPT(I)$ for any instance. The competitive ratio ρ_A of the on-line algorithm A is defined as

$$\rho_A = \inf_{\rho}\{\rho \ge 1 : A(I) \le \rho \cdot OPT(I), \forall I\}$$

Moreover, the lower bound of the on-line problem is L if there does not exist an algorithm with competitive ratio smaller than L. If the competitive ratio ρ_A of an algorithm A for this on-line problem matches the lower bound, i.e. $\rho_A = L$, we call the algorithm optimal, or best possible.

For on-line scheduling problem to minimize makespan on single unbounded batch machine, Deng et al. (2003) and Zhang et al. (2001) prove that no on-line algorithm can get competitive ratio less than $(\sqrt{5}+1)/2$ even if all jobs' processing times are same. They also independently give the same on-line algorithm with competitive ratio matching the lower bound. Poon and Yu (2005) provide a more general class of algorithms that also have the optimal competitive ratio for this case. Thus, the unbounded capacity case is finished.

For the bounded version, when all jobs have the same release date, the optimal schedule can be found by the FBLPT (Full Batch Longest Processing Times) rule provided by Bartholdi (Brucker et al. 1998). For the on-line version, Liu and Yu (2000) show the GRLPT algorithm is 2-competitive. Zhang et al. (2001) provide two 2-competitive on-line algorithms, H^B and MH^B. They also provide an optimal algorithm for the special case where there are only two distinct release times. Poon and Yu (2005) present a class of algorithms called FBLPT-based algorithms that contain the above three algorithms, and show that the algorithms are 2-competitive for any bounded capacity. They also give a 7/4-competitive algorithm for the case $B = 2$.

Hoogeveen and Vestjens (2000) first study the on-line scheduling problem with delivery time. After completion of processing jobs need to be delivered. The objective is to minimize the time by which all jobs have been delivered. For one non-batch machine $B = 1$, they show the lower bound is $(\sqrt{5}+1)/2$, and provide the optimal algorithm. For the unbounded batch machine version, Tian et al. (2007) give a 2-competitive on-line algorithm for general case. When all jobs' processing times are the same, they provide an optimal algorithm with competitive ratio $(\sqrt{5}+1)/2$. Yuan et al. (2009) consider an restricted model where every job's processing time is not smaller than its delivery time, and provide an on-line algorithm with competitive ratio $(\sqrt{5}+1)/2$ which is also optimal.

Sometimes we assume all jobs have an identical processing time for ease of exposition. However, it is more reasonable to assume jobs have their processing times being in one interval. In semiconductor manufacturing integrated circuits (jobs) are produced through the same technical processes. Therefore they have their burn-in times (processing times) in a not big interval. In this paper we assume that the processing times of all jobs are in the interval $[p, (1+\phi)p]$, where $\phi = (\sqrt{5}-1)/2$. In Sect. 2, we will give a class of on-line algorithms with competitive ratio $(\sqrt{5}+1)/2$ to minimize makespan on a bounded batch processing machine. In Sect. 3, we also provide a class of on-line algorithms with competitive ratio $(\sqrt{5}+1)/2$ to minimize maximum delivery time on an unbounded batch machine. For the two problems above, our algorithms are proved to be optimal.

Throughout the paper we use σ to denote the schedule produced by our algorithm and π to denote an optimal schedule. Let $U(t)$ be the set of all unscheduled jobs available at time t, and let $|U(t)|$ be the cardinality of $U(t)$. For any job set J, we use $r(J)$, $p(J)$, and $q(J)$, to denote the minimum release date, the largest processing time, and the largest delivery time of J, respectively. For any batch B, $S(B)$ represents its starting time.

2 Minimizing makespan on a bounded batch machine

In this section, we consider on-line scheduling problem on a bounded batch machine with the objective of minimizing the makespan under the assumption that the processing times of all jobs are drawn in $[p, (1+\phi)p]$. This problem can be expressed as $1|r_j, p_j \in [p, (1+\phi)p], B < +\infty, \text{on-line}|C_{\max}$, where B is the capacity of batch machine.

A batch is called *full* if it contain exactly B jobs. Otherwise, it is *non-full.*

Since $p_j \in [p, (1+\phi)p]$, it is better to put long jobs in one batch as many as possible. We choose to start a batch as soon as it is full. In addition, in order to obtain the optimal algorithm a non-full batch should not wait too long. Otherwise the competitive ratio will be too large if there is no job coming. Scheduler needs to decide how long a non-full batch should wait before it is processed. Based on waiting and FBLPT, we give the following algorithm.

Algorithm H^B

```
Step 0. If the machine is idle and $U(t)$ is not empty at time $t$, determine $r(t) = \min\{r_j | J_j \in U(t)\}$, $p(t) = \max\{p_j | J_j \in U(t)\}$ and $\alpha(t)$, where $\alpha(t) \in [\phi p(t), (1+\phi)r(t) + \phi p(t)]$. Otherwise, wait until the machine is idle and at least a job is available.
Step 1. If $|U(t)| \geq B$, select the longest $B$ jobs in $U(t)$ as a batch and schedule the batch.
Step 2. If $0 < |U(t)| < B$ then
  Step 2A. If $t \geq \alpha(t)$, then schedule jobs in $U(t)$ as a single batch and start this batch.
  Step 2B. If $t \leq \alpha(t)$, wait until $\alpha(t)$ or the next arrival.
Step 3. Goto step 0.
```

According to Algorithm H^B, a full batch will be processed immediately and a non-full batch can be started at t only when $t \geq \alpha(t)$. When the number of unscheduled jobs is smaller than B, idle times may be produced.

Theorem 1 *The competitive ratio of Algorithm H^B is $1 + \phi$.*

Proof We use σ to denote the schedule produced by Algorithm H^B and π to denote an optimal schedule. Let t be the minimum time such that there are no idle interval between t and $C_{\max}(\sigma)$. Assume that there are k batches $B_1, B_2, \dots, B_k$ in this interval $[t, C_{\max}(\sigma)]$. Without loss of generality, we assume the k batches are indexed in order of their non-decreasing starting times.

If $k = 1$, then $t \leq (1+\phi)r(B_1) + \phi p(B_1)$ and $C_{\max}(\pi) \geq r(B_1) + p(B_1)$. Therefore,

$$C_{\max}(\sigma) = t + p(B_1) \leq (1+\phi)(r(B_1) + p(B_1)) \leq (1+\phi)C_{\max}(\pi)$$

If $k \geq 2$, we consider two cases.

Case 1: In σ, all of the batches $B_1, B_2, \dots, B_{k-1}$ are full.

Since $p_j \in [p, (1+\phi)p]$ for each job J_j, we have $C_{\max}(\pi) \geq p(B_1) + (k-1)p$.

We consider the jobs whose release dates are no less than time t. Denote the set of these jobs by S. By Algorithm H^B, it is clear that jobs in S can form $k-1$ batches at least, and the longest job has its processing time not shorter than $p(B_2)$. So we have $C_{\max}(\pi) \geq t + p(B_2) + (k-2)p$.

In addition, $C_{\max}(\sigma) \leq t + p(B_1) + p(B_2) + (k-2)(1+\phi)p$. Thus,

$$C_{\max}(\sigma) - C_{\max}(\pi) \leq p(B_1) + (k-2)\phi p$$

We know that $(1+\phi)p(B_1) + (k-2)p \leq p(B_1) + (k-1)p \leq C_{\max}(\pi)$. Therefore,

$$C_{\max}(\sigma)/C_{\max}(\pi) \leq 1 + \phi$$

Case 2: There are some non-full batches in the batches $B_1, B_2, \dots, B_{k-1}$.

Denote the last non-full batch before $S(B_k)$ by B_j. For each batch B_i $(j \leq i \leq k)$, let J_i be the longest job. We have $C_{\max}(\pi) \geq S(B_j) + p_{j+1} + (k-j-1)p$, and

$$C_{\max}(\sigma) = S(B_j) + p_j + p_{j+1} + \cdots + p_k \leq S(B_j) + p_j + p_{j+1} + (k-j-1)(1+\phi)p$$

Thus

$$C_{\max}(\sigma) - C_{\max}(\pi) \leq p_j + (k-j-1)\phi p$$

Case 2.1: If J_j is started before $S(B_j)$ in π, then

$$C_{\max}(\pi) \geq p_j + (k-j)p \geq (1+\phi)p_j + (k-j-1)p$$

It follows from two above inequalities that $C_{\max}(\sigma) - C_{\max}(\pi) \leq \phi C_{\max}(\pi)$

Case 2.2: If J_j is started at or after $S(B_j)$ in π, then

$$C_{\max}(\pi) \geq S(B_j) + p_j + (k-j-1)p \geq (1+\phi)p_j + (k-j-1)p$$

Thus, $C_{\max}(\sigma) - C_{\max}(\pi) \leq \phi C_{\max}(\pi)$. □

According to the result in Zhang et al. (2001), the lower bound of $1|r_j, B < +\infty$, on-line$|C_{\max}$ is $1+\phi$ even when $p_j = 1$. Thus the lower bound of this problem can not be smaller than $1+\phi$. Therefore, this algorithm is best possible.

3 Minimizing maximum delivery time on an unbounded batch machine

In this section, we deal with the problem $1|r_j, p_j \in [p, (1+\phi)p], q_j, B = +\infty$, on-line$|L_{\max}$. The problem can be described as follows. We have one unbounded batch machine and sufficiently many vehicles. Every job has a release date, processing time and delivery time. Jobs arrive over time. After processing on the batch machine, jobs need to be delivered by vehicles. All jobs in one batch have the same completion time. Let C_j be the completion time on batch machine of J_j, and L_j be the time by which J_j has been delivered, i.e., $L_j = C_j + q_j$. Our goal is to minimize the time $L_{\max}$, by which all jobs have been delivered, i.e., $L_{\max} = \max_j\{L_j : L_j = C_j + q_j\}$.

According to the result in Zhang et al. (2001), when $p_j = 1$ and $q_j = 0$, this problem can be regarded as $1|r_j, p_j = 1, B = +\infty$, on-line$|C_{\max}$, whose lower bound is $1+\phi$. Thus, there is no on-line algorithm with competitive ratio less than $1+\phi$ for $1|r_j, p_j \in [p, (1+\phi)p], q_j, B = +\infty$, on-line$|L_{\max}$.

Similar to Sect. 2, we will provide a class of on-line algorithms with competitive ratio $1+\phi$. Since the machine's capacity is infinite, we can assign all unscheduled jobs in one batch. We just need to decide when to start unscheduled jobs. The algorithm is following.

Algorithm H^{∞}

```
Step 0. If the machine is idle and $U(t)$ is not empty at time $t$, determine $r(t) = \min\{r_j|J_j \in U(t)\}$, $p(t) = \max\{p_j|J_j \in U(t)\}$ and $\alpha(t)$, where $\alpha(t) \in [\phi p(t), (1+\phi)r(t) + \phi p(t)]$. Otherwise, wait until the machine is idle and at least a job is available.
Step 1. If $t \geq \alpha(t)$, then schedule jobs in $U(t)$ as a single batch.
Step 2. If $t < \alpha(t)$, wait until $\alpha(t)$ or the next arrival.
Step 3. Goto step 0.
```

We use σ to denote the schedule produced by Algorithm H^{∞} and π to denote an optimal schedule. Let J_l be the first job in σ such that $L_l = L_{\max}(\sigma)$. Let t be the minimum time such that there is no idle time in the time interval $[t, L_{\max}(\sigma) - q_l]$. Assume that there are k batches $B_1, B_2, \ldots, B_k$ in this interval. Without loss of generality, we assume the k batches are indexed in order of their non-decreasing starting times, i.e., $S(B_1) < S(B_2) < \cdots < S(B_k)$.

According to Algorithm H^{∞}, we can get three properties easily:

1. $S(B_i) \geq \phi p(B_i)$, for $1 \leq i \leq k$. One batch could start at t only when $t \geq \alpha(t) \geq \phi p(t) \geq \phi p(B_i)$.
2. $r(B_i) > S(B_{i-1})$, for $2 \leq i \leq k$. All jobs processing in one batch must arrive the time when the previous batch is started. Otherwise, they should be assigned to the previous batch.
3. $t = S(B_1) \leq (1+\phi)r(B_1) + \phi p(B_1)$.

Theorem 2 *The competitive ratio of the Algorithm H^∞ is $1+\phi$.*

Proof For each batch B_i in σ, denote by $J_i(r_i, p_i, q_i)$ and $J_i^*(r_i^*, p_i^*, q_i^*)$, the longest job and the job with the largest delivery time, respectively. (If the longest job has the largest delivery time, then $J_i = J_i^*$.) We have $p_i = p(B_i)$ and

$$L_{\max}(\sigma) = t + p_1 + p_2 + \cdots + p_k + q_k^*$$

We check all possible cases by discussing how many batches are processed in $[t, L_{\max}(\sigma) - q_l]$.

Case 1: There is only one batch in $[t, L_{\max}(\sigma) - q_l]$, i.e., $k = 1$. We have $L_{\max}(\sigma) = t + p_1 + q_1^* \le (1+\phi)(r(B_1) + p_1) + q_1^*$.

Case 1.1: If $q_1^* \ge p_1$, then

$$\frac{L_{\max}(\sigma)}{L_{\max}(\pi)} \le \frac{(1+\phi)(r(B_1)+p_1)+q_1^*}{r(B_1)+p_1^*+q_1^*} \le \frac{(1+\phi)(r(B_1)+p_1)+p_1}{r(B_1)+p_1+p_1^*} \le 1+\phi$$

where $p_1 \le (1+\phi)p_1^*$.

Case 1.2: If $q_1^* < p_1$,

(1) In π, if J_1 and J_1^* are not in the same batch, then $L_{\max}(\pi) \ge r(B_1) + p_1 + p_1^*$. Thus

$$\frac{L_{\max}(\sigma)}{L_{\max}(\pi)} \le \frac{(1+\phi)(r(B_1)+p_1)+q_1^*}{r(B_1)+p_1+p_1^*} \le \frac{(1+\phi)(r(B_1)+p_1)+p_1}{r(B_1)+p_1+p_1^*} \le 1+\phi$$

(2) In π, if J_1 and J_1^* are in the same batch, then $L_{\max}(\pi) \ge r(B_1) + p_1 + q_1^*$. Thus

$$\frac{L_{\max}(\sigma)}{L_{\max}(\pi)} \le \frac{(1+\phi)(r(B_1)+p_1)+q_1^*}{r(B_1)+p_1+q_1^*} \le 1+\phi$$

Case 2: There are more than one batches in $[t, L_{\max}(\sigma) - q_l]$, i.e., $k \ge 2$. We have $L_{\max}(\sigma) = S(B_{k-1}) + p_{k-1} + p_k + q_k^*$.

Case 2.1: If $q_k^* \ge p_k$, then

$$\begin{aligned}\frac{L_{\max}(\sigma)}{L_{\max}(\pi)} &\le \frac{S(B_{k-1})+p_{k-1}+p_k+q_k^*}{S(B_{k-1})+p_k^*+q_k^*} \le \frac{S(B_{k-1})+p_{k-1}+2p_k}{S(B_{k-1})+p_k^*+p_k} \\ &\le \frac{(1+\phi)p_{k-1}+2p_k}{\phi p_{k-1}+p_k+p_k^*}\end{aligned}$$

and

$$\begin{aligned}&[(1+\phi)p_{k-1}+2p_k] - (1+\phi)(\phi p_{k-1}+p_k+p_k^*) \\ &= \phi p_{k-1} + \phi^2 p_k - (1+\phi)p_k^* \\ &\le \phi(1+\phi)p + \phi^2(1+\phi)p - (1+\phi)p \\ &\le (1+\phi)p - (1+\phi)p = 0\end{aligned}$$

Therefore, $L_{\max}(\sigma)/L_{\max}(\pi) \le 1+\phi$.

Case 2.2: If $q_k^* < p_k$,

(1) In π, J_k and J_k^* are in the same batch, then $L_{\max}(\pi) \ge S(B_{k-1}) + p_k + q_k^*$. Thus, $L_{\max}(\sigma) - L_{\max}(\pi) \le p_{k-1}$.

  (a) In π, J_{k-1} is started before $S(B_{k-1})$. $L_{\max}(\pi) \ge p_{k-1} + p_k \ge (1+\phi)p_{k-1}$. Then

$$L_{\max}(\sigma) - L_{\max}(\pi) \le \phi L_{\max}(\pi)$$

  (b) In π, J_{k-1} is started at or after $S(B_{k-1})$. $L_{\max}(\pi) \ge S(B_{k-1}) + p_{k-1} \ge (1+\phi)p_{k-1}$. Then

$$L_{\max}(\sigma) \le (1+\phi)L_{\max}(\pi)$$

(2) In π, J_k and J_k^* are not in the same batch. $L_{\max}(\pi) \ge S(B_{k-1}) + p_k + p_k^*$. Then

$$\begin{aligned}\frac{L_{\max}(\sigma)}{L_{\max}(\pi)} &\le \frac{S(B_{k-1}) + p_{k-1} + p_k + q_k^*}{S(B_{k-1}) + p_k + p_k^*}\\ &\le \frac{S(B_{k-1}) + p_{k-1} + 2p_k}{S(B_{k-1}) + p_k + p_k^*} \le \frac{(1+\phi)p_{k-1} + 2p_k}{\phi p_{k-1} + p_k + p_k^*}\end{aligned}$$

Similar to case 2.1, we have $L_{\max}(\sigma)/L_{\max}(\pi) \le 1+\phi$.

Note that we have checked all possible schedules, the competitive ratio of the Algorithm H^∞ is $1+\phi$. □

Since there is no on-line algorithm with competitive ratio less than $1+\phi$ for the problem $1|r_j, p_j \in [p, (1+\phi)p], q_j, B = +\infty$, on-line$|L_{\max}$, Algorithms H^∞ is best possible.

4 Conclusion

In this paper, we study the problems $1|r_j, p_j \in [p, (1+\phi)p], B < +\infty$, on-line$|C_{\max}$ and $1|r_j, p_j \in [p, (1+\phi)p], q_j, B = +\infty$, on-line$|L_{\max}$. For each problem studied here, we provide a class of algorithms which are optimal. When the first problem is changed to $1|r_j, p_j \in [p, (1+\phi)p], q_j, B < +\infty$, on-line$|L_{\max}$, we think there exists an optimal algorithm based on FBLDT (Full Batch Longest Delivery Times). For the general case of first problem, Poon and Yu (2005) present a class of FBLPT-based algorithms which are 2-competitive. Although we conjecture that there exist the algorithms with competitive ratio less than 2, it is very difficult to find one. Further research is required.

Acknowledgements The authors would like to thank two anonymous referees whose comments helped a lot 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, Kovalyov MY, Potts CN, Tautenhahn T, van de Velde SL (1998) Scheduling a batching machine. J Sched 1:31–54

Deng X, Poon CK, Zhang Y (2003) Approximation algorithms in batch processing. J Comb Optim 7:247–257

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

Lee CY, Uzsoy R, MartinVega LA (1992) Efficient algorithms for scheduling semiconductor burn-in operations. Oper Res 40:764–775

Liu Z, Yu W (2000) Scheduling one batch processor subject to job release dates. Discrete Appl Math 105:129–136

Poon CK, Yu W (2005) A flexible on-line scheduling algorithm for batch machine with infinite capacity. Ann Oper Res 133:175–181

Poon CK, Yu W (2005) On-line scheduling algorithms for a batch machine with finite capacity. J Comb Optim 9:167–186

Tian J, Fu R, Yuan J (2007) On-line scheduling with delivery time on a single batch machine. Theory Comput Sci 374:49–57

Yuan J, Li S, Tian J, Fu R (2009) A best on-line algorithm for the single machine parallel-batch scheduling with restricted delivery times. J Comb Optim 17:206–213

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