Information Processing Letters 116 (2016) 526–531

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

Online parallel-batch scheduling to minimize total weighted completion time on single unbounded machine ☆

Yang Fang [a,b], Xiwen Lu [c,*]

[a] Shanghai Zhenhua Heavy Industries Co., Ltd., Shanghai, China
[b] Antai College of Economics and Management, Shanghai JiaoTong University, Shanghai, China
[c] Department of Mathematics, School of Science, East China University of Science and Technology, Shanghai, China

ARTICLE INFO

Article history:
Received 19 March 2015
Received in revised form 11 March 2016
Accepted 24 March 2016
Available online 29 March 2016
Communicated by X. Wu

Keywords:
Parallel-batch scheduling
Online algorithm
Analysis of algorithms
Unbounded machine
Total weighted completion time

ABSTRACT

The online parallel-batch scheduling problem on single unbounded machine to minimize total weighted job completion time is studied. For the general case of processing time, we give an online algorithm with competitive ratio $\sqrt{5}+1$. When all jobs have identical processing times, we provide an optimal online algorithm.

1. Introduction

In classical machine scheduling problems, a machine can process at most one job at a time. Motivated by the burn-in operations in the final testing stage of semiconductor manufacturing, Lee et al. [1] introduced parallel-batch machine scheduling problem. A parallel-batch machine is a machine that can process up to B jobs simultaneously as a batch. All jobs in one batch start and complete at the same time, and the processing time of one batch is equal to the longest processing time of the jobs in it. Once a batch starts to be processed, we can not stop it. The parallel-batch machine could be bounded $B < +\infty$, or unbounded $B = +\infty$.

In this paper, we consider the online parallel-batch scheduling problem on an unbounded parallel-batch machine to minimize total weighted job completion time. Jobs arrive overtime. Information of jobs and the number of jobs n are not known in advance. Once a job J_j arrives, its release time r_j, processing time p_j and weight w_j become known. As jobs arrive, we have to decide to process them or to wait for more information. Let C_j be the completion time of job J_j on the batch machine. Our goal is to minimize the total weighted completion time, i.e. $\sum w_j C_j$. This problem can be described as $1|r_j, B = +\infty, online|\sum w_j C_j$.

The standard measure of quality of online algorithms is competitive ratio. For minimum optimal problem, an online algorithm is called ρ-competitive if, for any instance, the cost output obtained by the online algorithm is at most ρ times the optimal off-line cost. The competitive ratio of an online algorithm is defined as the infimum of all values ρ. Moreover, if there is not an online algorithm with competitive ratio less than L for some problem, we call the lower bound of this problem is L. If the competitive ratio ρ of an algorithm matches the lower bound, i.e. $\rho = L$, we call the algorithm optimal, or best possible.

☆ This research was supported by National Natural Science Foundation of China (No. 11371137).

* Corresponding author.
E-mail address: xwlu@ecust.edu.cn (X. Lu).

http://dx.doi.org/10.1016/j.ipl.2016.03.010
0020-0190/

In last two decades, many of the researchers focused their attention on the objective functions of $\sum C_j$ and $\sum w_j C_j$. For the problem on single machine, Hoogeveen and Vestjens [2] showed that no deterministic online algorithm could have a competitive ratio less than 2 even if all jobs have identical weights. For the case of identical weights, Hoogeveen and Vestjens [2], and Lu et al. [3] presented several optimal algorithms with competitive ratio 2. For the general case, Anderson and Potts [4] obtained an optimal algorithm with competitive ratio 2.

On single parallel-batch machine, Chen et al. [5] gave a 10/3-competitive algorithm for the problem $1|r_j, B = +\infty, online|\sum w_j C_j$ by using the method of Greedy Interval, and presented a $(4+\epsilon)$-competitive algorithm for the bounded case $1|r_j, B < +\infty, online|\sum w_j C_j$. On m parallel-batch machines, Cao et al. [6] considered the online scheduling of equal length jobs with precedence constraint. For the unbounded batching version, they provided the lower bound and a best possible algorithm. For the bounded batching case with identical weights, they presented a 2-competitive algorithm.

There are several results for online overtime scheduling problems on single parallel-batch machine. Recently, Tian et al. [7] gave a literature review. When objective function is minimizing makespan, Deng et al. [8] and Zhang et al. [9] proved that no online algorithm could get its competitive ratio less than $(\sqrt{5}+1)/2$ even if all jobs' processing times are same. For the unbounded case, Deng et al. [8] and Zhang et al. [9] also independently gave the same online algorithm with competitive ratio matching the lower bound. Poon and Yu [10] provided a more general class of algorithms which also have optimal competitive ratio. Thus, the unbounded capacity case is finished. However, there is a long way to solve the bounded case. According to our literature reviewing, for the general case on single bounded machine, the known best competitive ratio is 2. Poon and Yu [11] presented a class of FBLPT (full batch longest processing time)-based algorithms with competitive ratio 2. They also obtained a 7/4-competitive algorithm for the case $B = 2$. When all jobs have their processing times in $[p, (1+\phi)p]$ ($p > 0$ and $\phi = (\sqrt{5}-1)/2$), Fang et al. [12] provided a class of optimal algorithms. Fu et al. [13] investigated the unbounded model with restarts. They proved that no online algorithm has a competitive ratio less than $(5-\sqrt{5})/2$, and they presented a 1.5-competitive algorithm. Later, Yuan et al. [14] designed a best possible online algorithm.

In this paper, we further consider the online batch parallel-scheduling problem $1|r_j, B = +\infty, online|\sum w_j C_j$. For the general case of processing time, we give an online algorithm with competitive ratio $\sqrt{5}+1$. When all jobs have equal processing times, an optimal algorithm is provided.

Throughout the paper, we denote by σ the schedule produced by the algorithm and by π an optimal schedule.

2. General case

For the problem $1|r_j, B = +\infty, online|\sum w_j C_j$, Chen et al. [5] designed an online algorithm according to the idea of Greedy Interval. In their algorithm, the start time of batch processing is set at $t = 2^i$ $(i = 0, 1, \cdots)$. Based on the idea of shift and interval, we can obtain a better algorithm and prove that the competitive ratio of the algorithm is $\sqrt{5}+1$.

Before giving the algorithm, we introduce some notations. Let $\tau = (1+\phi)/2 \approx 0.809$. Denote that $\tilde{r}_j = \max\{r_j, p_j\}$. When a job J_j arrives, we set $\tilde{r}_j$ before which the job is not available to be processed. Then for each job J_j considered to be processed at time t, we have $t \geq \tilde{r}_j$. Once the machine is idle at time t, we consider to process jobs. The time t is named *decision time*. Let $U(t)$ be the set of unscheduled jobs available at *decision time* t, satisfying $U(t) = \{J_j | \tilde{r}_j \in (t', t]\}$, where t' is the last decision time. Divide $U(t)$ into two parts, $X(t)$ and $Y(t)$,

$$X(t) = \{J_j | p_j \leq \phi t,\ J_j \in U(t)\},$$

$$Y(t) = \{J_j | p_j > \phi t,\ J_j \in U(t)\},$$

where $\phi = \frac{\sqrt{5}-1}{2} \approx 0.618$.

Let $C(t)$ be the job set chosen to be processed at time t, and $D(t)$ be the set of unscheduled jobs of $U(t)$. For any job set J, let $p(J)$ be the largest processing time in J. Let $W^X(t)$, $W^Y(t)$, $W(t)$ be the sum of job's weight in $X(t)$, $Y(t)$, $U(t)$, respectively. Without loss of generality, we assume that the first job arrives at zero and its processing time is greater than zero.

Algorithm H:

Step 0: Set $t = 0$, $t' = 0$, $D(t') = \emptyset$.
Step 1: If the machine is idle at time t, goto Step 2. Otherwise, wait.
Step 2: If at time t, $U(t) \neq \emptyset$, then goto Step 3. Otherwise, goto Step 4.
Step 3: Make decision by the ratio of $W^X(t)/W(t)$.

Step 3.1. If $W^X(t)/W(t) \geq \tau$, then let $C(t) = X(t) \cup D(t')$, and start to process $C(t)$ as one batch immediately. Set $t' := t$, $t := t + p(C(t))$, $D(t') = Y(t')$, and return to Step 2.

Step 3.2. If $W^X(t)/W(t) < \tau$, then let $C(t) = U(t) \cup D(t')$, and start to process $C(t)$ as one batch immediately. Set $t' := t$, $t := t + p(C(t))$, $D(t') = \emptyset$, and return to Step 2.

Step 4: If $D(t') \neq \emptyset$, process $D(t')$ as one batch immediately. Set $D(t') = \emptyset$, $t = t + p(D(t'))$, and return to Step 2. Otherwise, wait unit next time when $U(t) \neq \emptyset$, and return to Step 2.

Denote all decision times by $t_1, t_2, \cdots, t_k, \cdots$. Obviously, $U(t_k) = \{J_j | \tilde{r}_j \in (t_{k-1}, t_k]\}$, where $k \geq 1$ and $t_0 = 0$. Because every job is included in one of these job sets, $\sum w_j C_j(\sigma) = \sum_k \sum_{j \in U(t_k)} w_j C_j(\sigma)$. Thus we just need to analyze each $U(t_k)$. Let the three types of batches produced by Step 3.1, 3.2 and 4 of Algorithm H be T1, T2 and T3, respectively. Denote the batch starting to be processed at time t_k by $B_k(\sigma)$. Note that if the type of $B_k(\sigma)$ is T3, then $U(t_k) = \emptyset$.

Lemma 1. *If for any small positive number $\epsilon > 0$, the batch machine is idle at time $t_k - \epsilon$ in σ, then*