J Comb Optim (2011) 21: 138–149
DOI 10.1007/s10878-009-9231-z

Semi-online scheduling on 2 machines under a grade of service provision with bounded processing times

Ming Liu · Chengbin Chu · Yinfeng Xu · Feifeng Zheng

Published online: 23 May 2009

Abstract We study the problem of semi-online scheduling on 2 machines under a grade of service (GoS). GoS means that some jobs have to be processed by some machines to be guaranteed a high quality. The problem is online in the sense that jobs are presented one by one, and each job shall be assigned to a time slot on its arrival. Assume that the processing time p_i of every job J_i is bounded by an interval $[a, \alpha a]$, where $a > 0$ and $\alpha > 1$ are two constant numbers. By knowing the bound of jobs' processing times, we denote it by semi-online problem. We deal with two semi-online problems.

The first one concerns about bounded processing time constraint. First, we show that a lower bound of competitive ratio is: (1) $\frac{1+\alpha}{2}$ in the case where $1 < \alpha < 2$; (2) $\frac{3}{2}$ in the case where $2 \leq \alpha < 5$; and (3) $\frac{4+\alpha}{6}$ in the case where $5 \leq \alpha < 6$. We further propose an algorithm, called *B-ONLINE*, and prove that in the case where $\frac{25}{14} \leq \alpha$ and the optimal makespan $C_{OPT} \geq 20a$, *B-ONLINE* algorithm is optimal.

For the second problem, we further know the sum of jobs' processing times Σ in advance. We first show a lower bound $\frac{1+\alpha}{2}$ in the case where $1 < \alpha < 2$, then we

M. Liu (✉) · C. Chu · Y. Xu · F. Zheng
School of Management, Xi'an Jiaotong University, Xi'an, Shaanxi Province, 710049, P.R. China
e-mail: minyivg@gmail.com

C. Chu
e-mail: chengbin.chu@ecp.fr

Y. Xu
e-mail: yfxu@mail.xjtu.edu.cn

F. Zheng
e-mail: zhengff@mail.xjtu.edu.cn

M. Liu · C. Chu
Laboratoire Génie Industriel, Ecole Centrale Paris, Grande Voie des Vignes,
92295 Châtenay-Malabry Cedex, France

propose an algorithm *B-SUM-ONLINE* which is optimal in the case where $\Sigma \geq \frac{2\alpha}{\alpha-1}a$ and $1 < \alpha < 2$.

Keywords Online scheduling · Makespan · Competitive analysis · Grade of service · Bounded processing times · Total processing time

1 Introduction

Grade of service (GoS) is a qualitative concept, and it's often translated into the level of access privilege of different service provision. For example, suppose we have 2 machines (or processors). One of them can provide high quality service (or high GoS) while the other one provides normal service (or low GoS). Some jobs which request high quality must be processed by high GoS machine, while other jobs with low quality requests can be processed by both machines whenever they are available. For more recent development on GoS, see Hwang et al. (2004). The problem is *online* in the sense that when receiving a job and before the next job is presented, we must irrevocably assign the job to a time slot of the schedule. The assignment of each job shall be made on its arrival and the next job arrives immediately after the assignment, i.e., the difference of arrival times of two continual jobs is ignorable. Preemption and re-assignment are not allowed. The objective is to minimize the makespan, that is the completion time of the last job in the schedule. When all information is available at one time before scheduling, the problem is called *offline*. We call a problem semi-online if we know some information of jobs in advance, i.e., jobs' total processing time.

We use the competitive analysis (Borodin and El-Yaniv 1998) to measure the performance of an online algorithm. For any input job sequence I, let $C_{ON}(I)$ denote the makespan of the schedule produced by the online algorithm $\mathcal{A}_{ON}$ and $C_{OPT}(I)$ denote the makespan of the optimal schedule. We say that $\mathcal{A}_{ON}$ is ρ-competitive if

$$C_{ON}(I) \leq \rho C_{OPT}(I) + v$$

where v is a constant number. We also say that ρ is the competitive ratio of $\mathcal{A}_{ON}$. Semi-online algorithm is measured by the same way. We say that an algorithm is optimal if its competitive ratio matches the lower bound of competitive ratio.

Hwang et al. (2004) first study the (offline) problem of parallel machine scheduling with GoS eligibility. They proposed an approximation algorithm LG-LPT, and proved that its makespan is not greater than $\frac{5}{4}$ times the optimal makespan for $m = 2$ and not greater than $2 - \frac{1}{m-1}$ times the optimal makespan for $m \geq 3$. However, online scheduling under GoS eligibility was first studied by Park et al. (2006) and Jiang et al. (2006). For the problem of online scheduling on 2 machines with GoS constraint, they respectively proposed an optimal algorithm with a competitive ratio of $\frac{5}{3}$. Jiang (2008) further investigated the problem of online scheduling on parallel machines with two GoS levels. He assumed that the number of machines providing high GoS is not known before scheduling and decisions must be made without knowledge of the exact number of machines providing high GoS. I.e., we only know that in 10 parallel machines there are k ($1 \leq k \leq 9$) machines which can provide high GoS. Under this

consideration, he proved that 2 is a lower bound of online algorithms and proposed an online algorithm with a competitive ratio of $\frac{12+4\sqrt{2}}{7}$.

He and Zhang (1999) investigated two different semi on-line scheduling problems on a two-machine system. In the first problem, they assumed that all jobs have their processing times in between p and rp ($p > 0, r \geq 1$). They showed that *LS* is optimal with a competitive ratio $(r+1)/2$ in the case where $1 \leq r < 2$ and $3/2$ in the case where $r \geq 2$. In the second problem, they supposed that the largest processing time is known in advance. They showed that PLS algorithm is optimal with a competitive ratio $4/3$.

In this paper we will consider semi-online scheduling on two machines under a grade of service provision with jobs' processing times bounded by an interval $[a, \alpha a]$, where $a > 0$ and $\alpha > 1$ are two constant numbers. For simplicity, we use online algorithm to denote semi-algorithm in the remainder.

The rest of this paper is organized as follows. In Sect. 2.1, we will describe the problem and introduce some basic notations. Section 2.2 will show some lower bounds of competitive ratio considering different values of α. In Sect. 2.3, we will propose an algorithm *B-ONLINE*, and prove that in the case where $\frac{25}{14} \leq \alpha$ and the optimal makespan $C_{OPT} \geq 20a$, *B-ONLINE* is optimal. Finally, we will discuss the time complexity of *B-ONLINE* and give an illustrative example in Sect. 2.4. In Sect. 3.1, we will give some problem definitions and notations. Section 3.2 will present a lower bound of competitive ratio. In Sect. 3.3, we will show an algorithm *B-SUM-ONLINE* and prove that it is optimal in the case $\Sigma \geq \frac{2\alpha}{\alpha-1}a$ and $1 < \alpha < 2$. In Sect. 3.4, we will analyze the time complexity of *B-SUM-ONLINE* and give an illustrative example.

2 Online scheduling on 2 machines with GoS and bounded processing times

In this section, we study the problem of online scheduling on 2 machines under a grade of service provision with bounded processing times.

2.1 Problem definitions and notations

We are given 2 machines with speed of 1. Without loss of generality, we denote the one that can provide both high and low GoSs by machine-1, and the other one that only provides low GoS by machine-2. We denote each job by $J_i = (p_i, g_i)$, where p_i is the processing time of J_i and $g_i \in \{1, 2\}$ is the GoS of the job. $g_i = 1$ if J_i must be processed by machine-1 and $g_i = 2$ if it can be processed by either of the two machines. p_i and g_i are not known unless J_i arrives. A sequence of jobs $\sigma = \{J_1, J_2, \ldots, J_n\}$ which arrive online have to be scheduled irrevocably on one of the two machines on their arrivals. Each job J_i is presented immediately after J_{i-1} is scheduled. The schedule can be seen as a partition of job sequence σ into two subsequences, denoted by S_1 and S_2, where S_1 and S_2 consist of jobs assigned to machine-1 and machine-2, respectively. Let $L_1 = t(S_1) = \sum_{J_i \in S_1} p_i$ and $L_2 = t(S_2) = \sum_{J_i \in S_2} p_i$ denote the loads (or total processing times) of machine-1 and machine-2, respectively. Hence, the makespan of one schedule is $\max\{L_1, L_2\}$. The online problem can be written as:

$$\text{Given } \sigma, \text{ find } S_1 \text{ and } S_2 \text{ to minimize } \max\{L_1, L_2\}.$$

Let C_{ON} and C_{OPT} denote the makespan of online algorithm and offline optimal algorithm (for short, offline algorithm), respectively.

2.2 Lower bounds of competitive ratio

In this section, we will show some lower bounds of the competitive ratio of online algorithms for different values of α. If $\alpha = 1$, online algorithms can reach the optimal makespan. An algorithm is optimal if it assigns the jobs as many as possible on machine-2 when the difference of the loads of two machines is not greater than 1. Moreover, for the case where $\alpha \geq 6$, the lower bound of $\frac{5}{3}$ has been proved in Park et al. (2006). So, we will focus on the case where $1 < \alpha < 6$.

Theorem 1 *For the problem of online scheduling on two machines under GoS constraint with jobs' processing times bounded within interval* $[a, \alpha a]$, *there exists no algorithm with a competitive ratio less than*: (1) $\frac{1+\alpha}{2}$ *in the case where* $1 < \alpha < 2$; (2) $\frac{3}{2}$ *in the case where* $2 \leq \alpha < 5$; *and* (3) $\frac{4+\alpha}{6}$ *in the case where* $5 \leq \alpha < 6$.

Proof Without loss of generality, let $a = 1$. We will discuss the three cases of α in the following.

Case 1 $1 < \alpha < 2$. Let $\varphi = \frac{1+\alpha}{2}$. We will generate a job sequence ρ consisting of at most 3 jobs, which arrive one by one. Once the ratio of makespans between online and offline algorithms is at least φ after some job is assigned, no more jobs will be presented and we stop. We begin with job $J_1 = (1, 2)$. If online algorithm assigns J_1 to machine-1, we further generate job $J_2 = (1, 1)$. Since offline algorithm will assign J_1 to machine-2, $\frac{C_{ON}}{C_{OPT}} = 2 > \varphi$ and we stop. Otherwise if online algorithm assigns J_1 to machine-2, we generate job $J_2 = (1, 2)$. If J_2 is assigned to machine-2, we have $\frac{C_{ON}}{C_{OPT}} = 2 > \varphi$. Otherwise if J_2 is assigned to machine-1, we generate job $J_3 = (\alpha, 1)$. We have $\frac{C_{ON}}{C_{OPT}} = \frac{1+\alpha}{2} = \varphi$ in this case.

Case 2 $2 \leq \alpha < 5$. Similar to the analysis in Case 1, we begin with jobs $J_1 = J_2 = (1, 2)$. If online algorithm assigns both of them to one of the two machines, we have $\frac{C_{ON}}{C_{OPT}} = 2 > \frac{3}{2}$. Otherwise, we further generate job $J_3 = (2, 1)$. Then we have $C_{ON} = 3$ and $C_{OPT} = 2$, since the optimal solution consists of scheduling J_1, J_2 on machine-2 and J_3 on machine-1. It follows that $\frac{C_{ON}}{C_{OPT}} = \frac{3}{2}$.

Case 3 $5 \leq \alpha < 6$. Let $\varphi = \frac{4+\alpha}{6}$. We will generate a job sequence which consists of at most 5 jobs in this case. Similarly, we begin with jobs $J_1 = J_2 = (1, 2)$ and observe the behavior of online algorithm. If both of them are assigned to one of the two machines, $\frac{C_{ON}}{C_{OPT}} = 2 > \varphi$ and we stop. Otherwise, we further generate job $J_3 = (1, 2)$. If J_3 is assigned to machine-1, we give job $J_4 = (3, 1)$ and then $\frac{C_{ON}}{C_{OPT}} = \frac{1+1+3}{3} = \frac{5}{3} > \varphi$. Otherwise if J_3 is assigned to machine-2, we further generate job $J_4 = (3, 2)$. If J_4 is assigned to machine-2, it follows that $\frac{C_{ON}}{C_{OPT}} = \frac{1+1+3}{3} = \frac{5}{3} > \varphi$. Otherwise if J_4 is assigned to machine-1, we give the last job $J_5 = (\alpha, 1)$, and then $\frac{C_{ON}}{C_{OPT}} = \frac{1+3+\alpha}{6} = \frac{4+\alpha}{6} = \varphi$.

Therefore, the theorem follows. □

Remark 1 In the case where $\alpha \geq 6$, the tight lower bound of competitive ratio is $\frac{5}{3}$ (see Park et al. 2006).

2.3 *B-ONLINE* algorithm

Combining the *ONLINE* algorithm proposed in Park et al. (2006) with bounded jobs' processing time of job, we will give a modified algorithm called *B-ONLINE*. Before describing the algorithm, we give some notations on *B-ONLINE*'s schedules as follows.

At the arrival of each job, P and T are updated to become the maximum processing time and a half of total processing times of all arrived jobs, respectively. D is updated to be the total processing time of all arrived jobs with $g_i = 1$. Let $L = \max\{T, P, D\}$. Thus, we have $C_{OPT} \geq L$. For analysis convenience, we define $P^i, T^i, D^i, L^i, S_1^i$ and S_2^i to be the P, T, D, L, S_1 and S_2 respectively immediately after job J_i is assigned, and let $P^0 = T^0 = D^0 = L^0 = 0$ and $S_1^0 = S_2^0 = \emptyset$.

According to Theorem 1 on lower bounds, we define various values of parameter φ as follows. (1) $\varphi = \frac{1+\alpha}{2}$ in the case where $\frac{25}{14} \leq \alpha \leq 2$; (2) $\varphi = \frac{3}{2}$ in the case where $2 \leq \alpha \leq 5$; (3) $\varphi = \frac{4+\alpha}{6}$ in the case where $5 \leq \alpha < 6$; and (4) $\varphi = \frac{5}{3}$ in the case where $\alpha \geq 6$. *B-ONLINE* behaves as follows:

Step 1: Let $S_1 = \emptyset$, $S_2 = \emptyset$, $P = 0$, $T = 0$, $D = 0$;
Step 2: Receive job $J_i = (p_i, g_i)$. $P = \max\{P, p_i\}$ and $T = T + \frac{p_i}{2}$;
Step 3: If $g_i = 1$, let $S_1 = S_1 \cup \{J_i\}$ and $D = D + p_i$. Go to **Step 5**;
Step 4: Let $L = \max\{T, D, P\}$, if $t(S_2) + p_i \leq \varphi L$, let $S_2 = S_2 \cup \{J_i\}$; Else, let $S_1 = S_1 \cup \{J_i\}$;
Step 5: If no more jobs arrive, stop and output S_1 and S_2; Else, let $i = i + 1$ and go to **Step 2**.

B-ONLINE has the same performance in competitiveness as *ONLINE* in the case where $\alpha \geq 6$, i.e., both of them have a competitive ratio of $\frac{5}{3}$ in the case (please refer to Park et al. 2006 for details). So, we will focus our attention on the case where $\frac{25}{14} \leq \alpha < 6$ later on.

Lemma 1 *Given a constant number a, in the case where $\frac{25}{14} \leq \alpha < 6$, if job $J_i = (p_i, 2)$ is scheduled on machine-1 by* B-ONLINE *algorithm, there must be $t(S_1^i) < \frac{2-\varphi}{\varphi} t(S_2^i) + \frac{2\alpha a}{\varphi}$, where (1) $\varphi = \frac{1+\alpha}{2}$ in the case where $\frac{25}{14} \leq \alpha \leq 2$; (2) $\varphi = \frac{3}{2}$ in the case where $2 \leq \alpha \leq 5$; (3) $\varphi = \frac{4+\alpha}{6}$ in the case where $5 \leq \alpha < 6$.*

Proof If $J_i = (p_i, 2)$ is scheduled on machine-1, there must be $t(S_2^{i-1}) + p_i > \varphi L^i$. Since $t(S_2^i) = t(S_2^{i-1})$ and $p_i \in [a, \alpha a]$, we have $t(S_2^i) = t(S_2^{i-1}) > \varphi L^i - \alpha a$. In another aspect, because $L^i \geq T^i = \frac{1}{2}[t(S_1^i) + t(S_2^i)]$, we have

$$t(S_2^i) > \frac{\varphi}{2}[t(S_1^i) + t(S_2^i)] - \alpha a. \tag{1}$$

It follows that $t(S_1^i) < \frac{2-\varphi}{\varphi} t(S_2^i) + \frac{2\alpha a}{\varphi}$. The lemma follows. □

Theorem 2 *Given a constant number a, in the case where $\frac{25}{14} \le \alpha < 6$ and $C_{OPT} \ge 20a$*, B-ONLINE *is optimal with a competitive ratio φ such that* (1) $\varphi = \frac{1+\alpha}{2}$ *in the case where* $\frac{25}{14} \le \alpha \le 2$; (2) $\varphi = \frac{3}{2}$ *in the case where* $2 \le \alpha \le 5$; (3) $\varphi = \frac{4+\alpha}{6}$ *in the case where* $5 \le \alpha < 6$.

Proof We define C^i_{ON} and C^i_{OPT} to be C_{ON} and C_{OPT} respectively immediately after job J_i is scheduled. Thus, $C^i_{OPT} \ge C^{i-1}_{OPT}$ for $i \ge 1$.

We first assume that the theorem is false, implying that there must exist at least one instance, called *counter example*, which derives $\frac{C_{ON}}{C_{OPT}} > \varphi$. Among all such counter examples, let ς be the one with the least number n of jobs, called *minimal counter example*. By the definition of minimal counter example, the makespan of ς is not determined until the arrival of job J_n. Therefore,

$$C^n_{ON} = \max\{t(S^n_1), t(S^n_2)\} > \varphi C^n_{OPT}, \tag{2}$$

$$\max\{t(S^{n-1}_1), t(S^{n-1}_2)\} \le \varphi C^{n-1}_{OPT}. \tag{3}$$

Case 1 $g_n = 2$. If $J_n = (p_n, 2)$ is scheduled on machine-2, then $t(S^{n-1}_2) + p_n \le \varphi L^n \le \varphi C^n_{OPT}$. This implies that $\varphi C^n_{OPT} < C^n_{ON} = t(S^n_1) = t(S^{n-1}_1) \le \varphi C^{n-1}_{OPT}$. Since $C^n_{OPT} \ge C^{n-1}_{OPT}$, there is a contradiction. Therefore, J_n must be assigned to machine-1 and we have

$$t(S^{n-1}_2) + p_n > \varphi L^n. \tag{4}$$

Since $T^n = \frac{1}{2}[t(S^{n-1}_1) + t(S^{n-1}_2) + p_n] \le L^n$, together with inequality (4), we have $t(S^{n-1}_1) < (2-\varphi)L^n$. Since $L^n \le C^n_{OPT}$, we have $t(S^{n-1}_1) < (2-\varphi)C^n_{OPT}$. Considering $p_n \in [a, \alpha a]$, it follows $C^n_{ON} = t(S^n_1) = t(S^{n-1}_1) + p_n < (2-\varphi)C^n_{OPT} + \alpha a$. Since $C^n_{OPT} \ge 20a$, we have

$$\frac{C^n_{ON}}{C^n_{OPT}} = (2-\varphi) + \frac{\alpha a}{C^n_{OPT}} \le 2 - \varphi + \frac{\alpha}{20}.$$

To prove the theorem, we need to derive a contradiction to the assumption, i.e., to prove $\frac{C^n_{ON}}{C^n_{OPT}} \le \varphi$. That means $2 - \varphi + \frac{\alpha}{20} \le \varphi$ or $2 + \frac{\alpha}{20} - 2\varphi \le 0$.

Case 1.1 $\frac{25}{14} \le \alpha \le 2$. In this subcase, $\varphi = \frac{1+\alpha}{2}$, and then

$$2 + \frac{\alpha}{20} - 2\varphi = 2 + \frac{\alpha}{20} - 1 - \alpha = 1 - \frac{19\alpha}{20} < 0.$$

It contradicts to the assumption.

Case 1.2 $2 \le \alpha \le 5$. In this subcase, $\varphi = \frac{3}{2}$ and then

$$2 + \frac{\alpha}{20} - 2\varphi = 2 + \frac{\alpha}{20} - 3 = \frac{\alpha}{20} - 1 < -\frac{3}{4} < 0.$$

Again, there is a contradiction to the assumption.

Case 1.3 $5 \le \alpha < 6$. In this subcase, $\varphi = \frac{4+\alpha}{6}$.

$$2 + \frac{\alpha}{20} - 2\varphi = 2 + \frac{\alpha}{20} - \frac{4+\alpha}{3} = \frac{40 - 17\alpha}{60} \le -\frac{3}{4} < 0.$$

Again, there is a contradiction.

Case 2 $g_n = 1$. According to the definition of minimal counter example,

$$C^n_{ON} = t(S^n_1) > \varphi C^n_{OPT}. \tag{5}$$

Since C^n_{OPT} is at least the sum of processing times of jobs with $g_i = 1$, inequality (5) means that S_1 must contain at least one job, named J_k, with $g_k = 2$. Let J_K be the last job with $g_K = 2$ assigned to machine-1. Let $A_K(S_1)$ be the set of jobs assigned to machine-1 after J_K has been assigned to machine-1. Thus, $t(S^n_1) = t(S^k_1) + t(A_K(S_1))$. Since C^n_{OPT} cannot be less than the sum of processing times of jobs with $g_i = 1$, we have $t(A_K(S_1)) \le C^n_{OPT}$. Therefore, $t(S^n_1) \le t(S^k_1) + C^n_{OPT}$. Together with Lemma 1,

$$t(S^n_1) \le \frac{2-\varphi}{\varphi} t(S^k_2) + \frac{2\alpha a}{\varphi} + C^n_{OPT}. \tag{6}$$

Since the total processing time of jobs doesn't vary among different algorithms, we have $t(S^n_1) + t(S^n_2) \le 2C^n_{OPT}$. Considering inequality (5), we have $t(S^n_2) < (2-\varphi)C^n_{OPT}$. Since $t(S^k_2) \le t(S^n_2)$, together with inequality (6), we have

$$t(S^n_1) < \left(\frac{(2-\varphi)^2}{\varphi} + 1\right) C^n_{OPT} + \frac{2\alpha a}{\varphi}.$$

Therefore,

$$\frac{C^n_{ON}}{C^n_{OPT}} < \frac{(2-\varphi)^2}{\varphi} + 1 + \frac{2\alpha a}{\varphi C^n_{OPT}} \le \frac{(2-\varphi)^2}{\varphi} + 1 + \frac{\alpha}{10\varphi}.$$

Similar to Case 1, we need to derive a contradiction to the assumption to prove the theorem, i.e., to prove $\frac{C^n_{ON}}{C^n_{OPT}} \le \varphi$. That means $\frac{(2-\varphi)^2}{\varphi} + 1 + \frac{\alpha}{10\varphi} \le \varphi$ or $4 + \frac{\alpha}{10} - 3\varphi \le 0$.

Case 2.1 $\frac{25}{14} \le \alpha \le 2$. In this subcase, $\varphi = \frac{1+\alpha}{2}$. Therefore,

$$4 + \frac{\alpha}{10} - 3\varphi = 4 + \frac{\alpha}{10} - \frac{3(1+\alpha)}{2} = \frac{25 - 14\alpha}{10} \le 0.$$

It contradicts to the assumption.

Case 2.2 $2 \le \alpha \le 5$. In this subcase, $\varphi = \frac{3}{2}$, and then

$$4 + \frac{\alpha}{10} - 3\varphi = 4 + \frac{\alpha}{10} - \frac{9}{2} = \frac{\alpha - 5}{10} < 0.$$

Again, there is a contradiction.

Case 2.3 $5 \leq \alpha < 6$. In this subcase, $\varphi = \frac{4+\alpha}{6}$ and thus

$$4 + \frac{\alpha}{10} - 3\varphi = 4 + \frac{\alpha}{10} - \frac{4+\alpha}{2} = \frac{2(5-\alpha)}{5} \leq 0.$$

There is a contradiction to the assumption.

According to the analysis in Cases 1 and 2, the theorem follows. □

2.4 Time complexity and an illustrative example

Assume that the number of jobs in an instance is n. For this instance, *B-ONLINE* algorithm executes n times. The Step 2 of *B-ONLINE* algorithm contains only one comparison operation. In Steps 3 and 4 of *B-ONLINE*, there are at most 4 comparisons. Therefore, in each iteration there are at most 5 comparisons and the time complexity of *B-ONLINE* is $O(n)$.

An example Given a job sequence $\{J_1 = (4, 2),\ J_2 = (3, 1),\ J_3 = (2, 2),\ J_4 = (4, 1),\ J_5 = (3, 2),\ J_6 = (4, 2),\ J_7 = (4, 2),\ J_8 = (3, 1),\ J_9 = (4, 2),\ J_{10} = (2, 2),\ J_{11} = (4, 1),\ J_{12} = (3, 2)\}$, the optimal algorithm can assign $J_2, J_4, J_8, J_9, J_{10}, J_{11}$ to machine-1 and the others to machine-2. Therefore, we have $C_{OPT} = 20$. We set $a = 1$ and $\alpha = 4$ is bounded by the interval $[2, 5]$. In this case, we have $\varphi = \frac{3}{2}$. *B-ONLINE* algorithm schedules $J_1, J_2, J_4, J_8, J_{11}$ on machine-1 and the others on machine-2. Thus, we have $C_{ON} = 22$ and $\frac{C_{ON}}{C_{OPT}} = \frac{22}{20} < \frac{3}{2} = \varphi$.

3 Online scheduling on 2 machines with the total processing time

In this section, we further know jobs' total processing time in advance. We also study the problem of online scheduling on 2 machines with GoS and bounded processing times.

3.1 Problem definitions and notations

We are given 2 machines with speed of 1 and jobs' total processing time Σ. Without loss of generality, we denote the machine that can provide both high and low GoSs by machine-1, and the other one that only provides low GoS by machine-2. We denote each job by $J_i = (p_i, g_i)$, where p_i is the processing time of J_i and $g_i \in \{1, 2\}$ is the GoS of the job. $g_i = 1$ if J_i must be processed by machine-1 and $g_i = 2$ if it can be processed by either of the two machines. p_i and g_i are not known unless J_i arrives. A sequence of jobs $\sigma = \{J_1, J_2, \ldots, J_n\}$ which arrive online have to be scheduled irrevocably on one of the two machines on their arrivals. Each job J_i is presented immediately after J_{i-1} is scheduled. The schedule can be seen as a partition of job sequence σ into two subsequences, denoted by S_1 and S_2, where S_1 and S_2 consist of jobs assigned to machine-1 and machine-2, respectively. Note that $\Sigma = t(S_1) + t(S_2)$. Let $L_1 = t(S_1) = \sum_{J_i \in S_1} p_i$ and $L_2 = t(S_2) = \sum_{J_i \in S_2} p_i$ denote the loads (or total

processing times) of machine-1 and machine-2, respectively. Hence, the makespan of one schedule is $\max\{L_1, L_2\}$. The online problem can be written as:

$$\text{Given } \sigma, \text{ find } S_1 \text{ and } S_2 \text{ to minimize } \max\{L_1, L_2\}.$$

Let C_{S-ON} and C_{OPT} denote the makespan of online algorithm and offline optimal algorithm (for short, offline algorithm), respectively.

3.2 Lower bounds of competitive ratio

In this section, we will show some lower bounds of competitive ratio of online algorithm considering difference values of α. If $\alpha = 1$, online algorithms can reach the optimal makespan. I.e., one algorithm is optimal if it assigns the jobs as many as possible on machine-2 when the difference of the loads of two machines is not greater than 1. Moreover, for the case where $\alpha \geq 2$, the lower bound of $\frac{3}{2}$ has been proved in Park et al. (2006). So, we will focus on the case where $1 < \alpha < 2$.

Theorem 3 *For the problem of online scheduling on two machines under GoS constraint with* (1) *knowledge of total processing time and* (2) *jobs' processing times bounded within interval* $[a, \alpha a]$, *there exists no online algorithm with a competitive ratio less than* $\frac{1+\alpha}{2}$ *in the case where* $1 < \alpha < 2$.

Proof Without loss of generality, let $a = 1$ and the sum of jobs' processing times be $2 + \alpha$. We will generate a job sequence which consists of at most 3 jobs, which arrive one by one. Once the ratio of makespans between online and offline algorithms is at least $1 + \frac{1}{\alpha}$ after some job is assigned, no more jobs will be presented and we stop. We begin with job $J_1 = (1, 2)$. If online algorithm assigns J_1 to machine-1, we further generate job $J_2 = (\alpha, 1)$ and $J_3 = (1, 2)$. Since offline algorithm will assign J_1, J_3 to machine-2 and J_2 to machine-1, it follows $\frac{C_{ON}}{C_{OPT}} \geq \frac{1+\alpha}{2}$ and we stop. Otherwise if online algorithm assigns J_1 to machine-2, we generate job $J_2 = (\alpha, 2)$ and $J_3 = (1, 1)$. If J_2 is assigned to machine-2, since offline algorithm will scheduling J_1, J_3 on machine-1 and J_2 on machine-2, we have $\frac{C_{ON}}{C_{OPT}} \geq \frac{1+\alpha}{2}$. Otherwise if J_2 is assigned to machine-1, we also generate job $J_3 = (1, 1)$. We have $\frac{C_{ON}}{C_{OPT}} = \frac{1+\alpha}{2}$ in this case, since offline algorithm will assign J_1, J_3 on machine-1 and J_2 on machine-2. □

Remark 2 In the case where $\alpha \geq 2$, the tight lower bound of competitive ratio is $\frac{3}{2}$ (see Park et al. 2006).

3.3 *B-SUM-ONLINE* algorithm

Combining the *SEMI-ONLINE* algorithm proposed in Park et al. (2006) with bounded jobs' processing times, we will give a modified algorithm called *B-SUM-ONLINE*. Before describing the algorithm, we give some notations on *B-SUM-ONLINE*'s schedules as follows.

Let Σ be the sum of jobs' processing times. Let $L = \frac{\Sigma}{2}$. Thus, we have $C_{OPT} \geq L$. *B-SUM-ONLINE* behaves as follows:

Step 1: Let $S_1 = \emptyset$, $S_2 = \emptyset$;
Step 2: Receive job $J_i = (p_i, g_i)$;
Step 3: If $g_i = 1$, let $S_1 = S_1 \cup \{J_i\}$. Go to **Step 5**;
Step 4: If $t(S_2) + p_i \leq \frac{1+\alpha}{2}L$, let $S_2 = S_2 \cup \{J_i\}$; Else, let $S_1 = S_1 \cup \{J_i\}$;
Step 5: If no more jobs arrive, stop and output S_1 and S_2; Else, let $i = i + 1$ and go to **Step 2**.

B-SUM-ONLINE has the same performance in competitiveness as *SEMI-ONLINE* in the case where $\alpha \geq 2$, i.e., both of them have competitive ratio of $\frac{3}{2}$ in the case (please refer to Park et al. 2006 for details). So, we will focus our attention on the case where $1 < \alpha < 2$ later on.

The proof of the competitive ratio of B-SUM-ONLINE algorithm is by contradiction. We assume that there exists a job instance, called counter example, for which B-SUM-ONLINE algorithm yields a schedule with makespan greater than $\frac{1+\alpha}{2}$ times of the optimum. We further define the counter example with the least number of jobs as minimal counter example. For a minimal counter example $\mathcal{I}$, let $\mathcal{J}_1 = \{J_i | g_i = 1, J_i \in \mathcal{I}\}$ and $\mathcal{J}_2 = \{J_i | g_i = 2, J_i \in \mathcal{I}\}$. We use σ to denote the schedule generated by B-SUM-ONLINE algorithm for $\mathcal{I}$. Let C_{S-ON} and C_{OPT} denote the makespan of B-SUM-ONLINE algorithm and the optimal algorithm for $\mathcal{I}$, respectively. We define S_1^j and S_2^j to be S_1 and S_2 that we have immediately after we schedule job J_j. $S_1^0 = S_2^0 = \emptyset$.

Lemma 2 $t(\mathcal{J}_2) > \frac{1+\alpha}{2}L$.

Proof Suppose $t(\mathcal{J}_2) \leq \frac{1+\alpha}{2}L$. For any job $J_j = (p_j, 2)$, since $t(S_2^{j-1}) + p_j \leq t(\mathcal{J}_2)$ and $t(\mathcal{J}_2) \leq \frac{1+\alpha}{2}L$, we have $t(S_2^{j-1}) + p_j \leq \frac{1+\alpha}{2}L$. From Step 4 of the algorithm, we get $S_1 = \mathcal{J}_1$ and $S_2 = \mathcal{J}_2$. Since $t(S_1) = t(\mathcal{J}_1) \leq C_{OPT}$, we know $C_{S-ON} \neq t(S_1)$. Therefore, $C_{S-ON} = t(S_2) = t(\mathcal{J}_2) \leq \frac{1+\alpha}{2}L \leq \frac{1+\alpha}{2}C_{OPT}$. There exists a contradiction. □

Theorem 4 *Given* $\Sigma \geq \frac{2\alpha}{\alpha-1}a$, *B-SUM-ONLINE algorithm is optimal with a competitive ratio of* $\frac{1+\alpha}{2}$ *in the case* $1 < \alpha < 2$.

Proof Let $\varphi = \frac{1+\alpha}{2}$. We assume that the theorem is false and there exists a minimal counter example $\mathcal{I} = \{J_1, \ldots, J_n\}$. Therefore, we have

$$C_{S-ON}^n = \max\{t(S_1^n), t(S_2^n)\} > \varphi C_{OPT}^n, \tag{7}$$

$$\max\{t(S_1^{n-1}), t(S_2^{n-1})\} \leq \varphi C_{OPT}^{n-1}. \tag{8}$$

Our aim is to prove for this instance, $\frac{C_{S-ON}^n}{C_{OPT}^n} \leq \varphi$ holds.

Case 1 $g_n = 2$. If J_n is assigned to machine-2, we have $t(S_2^{n-1}) + p_n \leq \varphi L \leq \varphi C_{OPT}^n$ and $t(S_1^{n-1}) = t(S_1^n)$. By inequality (7), it follows that $t(S_1^{n-1}) > \varphi C_{OPT}^n \geq \varphi C_{OPT}^{n-1}$. This contradicts inequality (8).

So J_n must be assigned to machine-1, which implies $t(S_2^{n-1}) + p_n > \varphi L$, $C_{S-ON}^n = t(S_1^n) > \varphi C_{OPT}^n$ and $t(S_2^n) = t(S_2^{n-1})$. Since $t(S_1^n) + t(S_2^n) = 2L$ and $C_{OPT}^n \geq L$,

$$C_{S-ON}^n - C_{OPT}^n \leq t(S_1^n) - L = L - t(S_2^n) = L - t(S_2^{n-1}). \tag{9}$$

Since $t(S_2^{n-1}) + p_n > \varphi L$, together with the above inequality,

$$C_{S-ON}^n - C_{OPT}^n < L - (\varphi L - p_n) = p_n - (\varphi - 1)L. \tag{10}$$

If $p_n > (\alpha - 1)L$, considering $p_n \in [a, \alpha a]$, it follows $\alpha a > (\alpha - 1)L$. This implies $L < \frac{\alpha}{\alpha-1}a$. Since $\Sigma = 2L$, we have $\Sigma < \frac{2\alpha}{\alpha-1}a$. This contradicts the assumption $\Sigma \geq \frac{2\alpha}{\alpha-1}$. So

$$p_n \leq (\alpha - 1)L. \tag{11}$$

Together with inequality (10), it follows that

$$C_{S-ON}^n - C_{OPT}^n < (\alpha - 1)L - (\varphi - 1)L < \frac{\alpha - 1}{2}L = (\varphi - 1)L < (\varphi - 1)C_{OPT}^n.$$

Therefore, $\frac{C_{S-ON}^n}{C_{OPT}^n} < \varphi$, there exists a contradiction.

Case 2 $g_n = 1$. From the minimality, we have $C_{S-ON}^n = t(S_1^n)$. By Lemma 2, we know that there is at least one job in $\mathcal{J}_2$ scheduled on machine-1. Otherwise, we have $\frac{C_{S-ON}^n}{C_{OPT}^n} < \varphi$ and there exists a contradiction. Let $J_k = (p_k, 2)$ denote the last job with $g_i = 2$ in $\mathcal{I}$ scheduled on machine-1, i.e., $\mathcal{I} = \{J_1, \ldots, J_{k-1}, J_k, J_{k+1}, \ldots, J_n\}$. In this case, by taking out J_k and putting it at last position in $\mathcal{I}$, we get a new instance $\mathcal{I}'$, i.e., $\mathcal{I}' = \{J_1, \ldots, J_{k-1}, J_{k+1}, \ldots, J_n, J_k\}$. Note that for this new instance $\mathcal{I}'$, the performance of B-SUM-ONLINE algorithm does not become worse, by Step 4 of the algorithm. Now, we renew the indexes of jobs in $\mathcal{I}'$ by their positions, i.e., $\mathcal{I}' = \{J_1' = J_1, \ldots, J_{k-1}' = J_{k-1}, J_k' = J_{k-1}, \ldots, J_{n-1}' = J_n', J_n' = J_k\}$. Since $g_n' = 1$, we have the result of *Case 1*.

Therefore, the theorem follows. □

3.4 Time complexity and an illustrative example

Assume that the number of jobs in an instance is n. For this instance, *B-SUM-ONLINE* algorithm executes n times. The Step 2 of *B-SUM-ONLINE* algorithm contains only one comparison operation. In Steps 3 and 4 of *B-SUM-ONLINE*, there are at most one comparison. Therefore, in each iteration there are at most 5 comparisons and the time complexity of *B-SUM-ONLINE* is $O(n)$.

An example Given a job sequence $\{J_1 = (1, 2),\ J_2 = (1.5, 2),\ J_3 = (1, 2),\ J_4 = (1.5, 1),\ J_5 = (1.5, 2),\ J_6 = (1.5, 2)\}$, the optimal algorithm can assign J_1, J_2, J_4 to machine-1 and the others to machine-2. Therefore, we have $C_{OPT} = 4$. We set $a = 1$ and $\alpha = 1.5$. In this case, we have $\varphi = \frac{1+\alpha}{2} = \frac{5}{4}$. *B-SUM-ONLINE* algorithm schedules J_1, J_2, J_3, J_5 on machine-2 and the others on machine-1. Thus, we have $C_{ON} = 5$ and $\frac{C_{ON}}{C_{OPT}} = \frac{5}{4} = \varphi$.

Acknowledgements The authors would like to thank Yiwei Jiang for communication and the anonymous referees for their helpful comments. This work is partially supported by NSF of China under Grants 70525004, 60736027 and 70702030.

References

Borodin A, El-Yaniv R (1998) Online computation and competitive analysis. Cambridge University Press, Cambridge

He Y, Zhang G (1999) Semi on-line scheduling on two identical machines. Computing 62:179–187

Hwang H, Chang S, Lee K (2004) Parallel machines scheduling under a grade of service provision. Comput Oper Res 31:2055–2061

Jiang Y (2008) Online scheduling on parallel machines with two GoS levels. J Comb Optim 16:28–38

Jiang Y, He Y, Tang C (2006) Optimal online algorithms for scheduling on two identical machines under a grade of service. J Zhejiang Univ Sci A 7(3):309–314

Park J, Chang SY, Lee K (2006) Online and semi-online scheduling of two machines under a grade of service provision. Oper Res Lett 34:692–696