Optim Lett (2016) 10:207–213
DOI 10.1007/s11590-014-0838-3

SHORT COMMUNICATION

Optimal algorithm for semi-online scheduling on two machines under GoS levels

Taibo Luo · Yinfeng Xu

Received: 28 February 2014 / Accepted: 18 November 2014 / Published online: 29 November 2014

Abstract Recently, Liu et al. (J Combin Optim 21:138–149, 2011) studied the semi-online scheduling problem on two machines under a grade of service provision. As the sum of jobs' processing times Σ is known in advance and the processing times are bounded by an interval $[1, \alpha]$ where $1 < \alpha < 2$, they presented an algorithm which is $\frac{1+\alpha}{2}$-competitive when $\Sigma \geq \frac{2\alpha}{\alpha-1}$. In this paper, we give a modified algorithm which is shown to be optimal for arbitrary α and Σ.

Keywords Online scheduling · Grade of service · Bounded processing times · Total processing time · Algorithms

1 Introduction

Scheduling problem under a grade of service (GoS) provision can be described as follows. We are given n independent jobs and m identical machines. Each job has a processing time and is labeled with a GoS level. Each machine is also labeled with a GoS level and a machine is allowed to process a job only when the GoS level of the machine is not greater than that of the job. The objective is to find a schedule which minimizes the makespan. This problem was first proposed by Hwang et al. [2] and they presented an LG- LPT algorithm which has a tight bound of $\frac{5}{4}$ for two machines

T. Luo · Y. Xu (✉)
Business School, Sichuan University, Chengdu 610065, China
e-mail: yfxu@scu.edu.cn

T. Luo
e-mail: luotaibo@126.com

Y. Xu
State Key Lab for Manufacturing Systems Engineering, Xi'an 710049, China

and a tight bound of $2 - \frac{1}{m-1}$ for m ($m \geq 3$) machines. Ji and Cheng [4] proposed a fully polynomial-time approximation scheme (FPTAS) for this problem. Woeginger [8] gave two FPTASs which are simpler compared with the FPTAS in [4]. For the online version of the scheduling problem under GoS levels, Park et al. [5] and Jiang et al. [3] independently presented an optimal algorithm with a competitive ratio of $\frac{5}{3}$ for the case of two machines.

For the case with unit processing time, Luo et al. [6] considered two semi-online models with two GoS levels on m parallel machines. The first model is a lookahead version where an online algorithm is able to foresee the information of the next k jobs. The second model is a buffer version where a buffer is available for storing at most g jobs. For the both models, The authors presented an optimal online algorithm with a competitive ratio of $\frac{m^2}{m(m-s)+s^2}$ for $k = \frac{m^2-1}{s} + s - m$ and an optimal online algorithm with a competitive radio of $\frac{m^2}{m(m-s)+s^2}$ for $g = m - \frac{m}{m(m-s)+s^2}$. s is the number of machines with high GoS level. Moreover, for the case where $m = 2$, they proved that the two algorithms can get their best possible competitive ratio of $\frac{4}{3}$ when $k = 1$ and $g = 1$, respectively.

However, for the semi-online version of the scheduling problem under GoS levels, most researches focus on the case of two machines. Park et al. [7] gave an optimal algorithm with a competitive ratio of $\frac{3}{2}$ when the total processing time of all jobs is known. Wu et al. [9] presented an optimal algorithm with a competitive ratio of $\frac{\sqrt{5}+1}{2}$ when the largest processing time of all jobs is known. And when the optimal value of the instance is known, they gave an optimal algorithm with a competitive ratio of $\frac{3}{2}$ in the same paper. Chen et al. [1] considered a semi-online scheduling on two machines under GoS levels with buffer or rearrangements and presented two optimal algorithms with a competitive ratio of $\frac{3}{2}$. Liu et al. studied two semi-online scheduling problems under GoS levels in [5]. The first problem is concerned with bounded processing time constraints, i.e., the processing time p_i is bounded by an interval $[1, \alpha]$ where $\alpha > 1$. The second problem assumes that, in addition to the bounded processing time constraints, the total processing time of all jobs is known in advance. For these two problems, they showed some lower bounds of the competitive ratio for different values of α. They also proposed two algorithms which are shown to be competitive for some situations. For the first problem studied by Liu et al. [5], Zhang et al. [10] improved the result and gave an optimal algorithm for arbitrary $\alpha \geq 1$.

In this paper, we focus on the second semi-online scheduling problem studied by Liu et al. [5]. For the case $\alpha = 1$, an online algorithm can reach the optimal makespan if it always schedules the job with $GoS = 2$ on the second machine until the total processing time of the jobs scheduled on the second machine is equal to $\lceil \frac{\Sigma}{2} \rceil$. As Park et al. [7] proved a lower bound of $\frac{3}{2}$ for the semi-online scheduling problem with known total processing time by using an example where the processing times are bounded in interval [1,2], the lower bound of competitive ratio is $\frac{3}{2}$ for the case $\alpha \geq 2$, and an optimal algorithm was presented by Park et al. [7]. For the case $1 < \alpha < 2$, a lower bound of $\frac{1+\alpha}{2}$ was proved by Liu et al. [5], and they also presented an algorithm B- SUM- ONLINE which is $\frac{1+\alpha}{2}$-competitive when $\Sigma \geq \frac{2\alpha}{\alpha-1}$. This paper modifies algorithm B- SUM- ONLINE and gets a new algorithm which is shown to be optimal for arbitrary α and Σ.

2 Definitions

We are given two machines and a series of jobs arriving online which are to be scheduled irrevocably at the time of their arrivals. The first machine can process all the jobs while the second one can process only part of the jobs. The arrival of a new job occurs only after the current job is scheduled. Let $\sigma = \{J_1, \ldots, J_n\}$ be the set of all jobs arranged in the order of arrival. We denote each job by $J_i = (p_i, g_i)$, where p_i is the processing time of job J_i and $g_i \in \{1, 2\}$ is the GoS level of job J_i. $g_i = 1$ if job J_i must be processed by the first machine, and $g_i = 2$ if it can be processed by either of the two machines. p_i and g_i are not known until the arrival of job J_i. Each p_i is bounded by an interval $[1, \alpha]$ where $\alpha \geq 1$ is a constant number, and the total processing time of all jobs Σ is known in advance.

The schedule can be seen as a partition of σ into two subsets, denoted by $\langle S_1, S_2\rangle$, where S_1 and S_2 contain job indices assigned to the first and the second machine, respectively. Let $t(S_1) = \Sigma_{J_i \in S_1} p_i$ and $t(S_2) = \Sigma_{J_i \in S_2} p_i$ denote the load of the first machine and the load of the second machine. Note that $t(S_1) + t(S_2) = \Sigma$. The maximum value of $t(S_1)$ and $t(S_2)$, i.e. $max\{t(S_1), t(S_2)\}$, is defined as the makespan of the schedule $\langle S_1, S_2\rangle$. The objective is to find a schedule $\langle S_1, S_2\rangle$ that minimizes the makespan.

Let C_{opt} be the minimum makespan obtained by an optimal off-line algorithm and C_A be the makespan generated by algorithm A. Then the competitive ratio of algorithm A is defined to be the supremum of the fraction $\frac{C_A}{C_{opt}}$. Let $L = \frac{\Sigma}{2}$. According to the definition, we have $C_{opt} \geq L$.

3 An optimal online algorithm

In this section, we present a modified algorithm which is shown to be not only optimal for the case $1 \leq \alpha < 2$ but also optimal for the case $\alpha \geq 2$ based on B- SUM- ONLINE. For convenience, we define S_1^i and S_2^i as S_1 and S_2 that we get immediately after scheduling job J_i. According to the lower bounds of competitive ratio, we define various values of parameter β as follows: (1) $\beta = \frac{1+\alpha}{2}$ when $1 \leq \alpha < 2$; (2) $\beta = \frac{3}{2}$ when $\alpha \geq 2$. First, we describe our algorithm as follows.

Algorithm M:
Step 1. Let $S_1^0 = \emptyset$, $S_2^0 = \emptyset$, $i = 1$;
Step 2. Receive job $J_i = (p_i, g_i)$;
Step 3. If $g_i = 1$, let $S_1^i = S_1^{i-1} \bigcup\{J_i\}$. Go to ***Step 5***;
Step 4. If $g_i = 2$,
4.1. If $t(S_2^{i-1}) + p_i \leq \beta L$, let $S_2^i = S_2^{i-1} \bigcup\{J_i\}$. Go to ***Step 5***;
4.2. **(Stopping criterion 1)**. If $t(S_2^{i-1}) + p_i > \beta L$ and $\Sigma - t(S_2^{i-1}) \leq \beta L$, assign job J_i and all the remaining jobs to S_1. Stop and output S_1 and S_2.

4.3. **(Stopping criterion 2)**. If $t(S_2^{i-1}) + p_i > \beta L$, $\Sigma - t(S_2^{i-1}) > \beta L$ and $\Sigma - t(S_2^{i-1}) - p_i < t(S_2^{i-1})$, assign job J_i and all the remaining jobs to S_1. Stop and output S_1 and S_2.
4.4. **(Stopping criterion 3)**. If $t(S_2^{i-1}) + p_i > \beta L$, $\Sigma - t(S_2^{i-1}) > \beta L$ and $\Sigma - t(S_2^{i-1}) - p_i \geq t(S_2^{i-1})$, assign job J_i to S_2 and assign all the remaining jobs to S_1. Stop and output S_1 and S_2.
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***.

Before proving algorithm M is optimal, we give a lemma first.

Lemma 1 *Suppose that* $1 \leq \alpha < 2$. *For an arbitrary job sequence* σ, *if the number of the jobs in* σ, *denoted by* n, *is an even number, then the total processing time of arbitrary* $\frac{n}{2}$ *jobs in* σ *is at most* $\frac{1+\alpha}{2}L$.

Proof Let S_h be a set of arbitrary $\frac{n}{2}$ jobs in σ, and S_l be the set of the other $\frac{n}{2}$ jobs. Define $t(S_h) = \Sigma_{J_i \in S_h} p_i$ and $t(S_l) = \Sigma_{J_i \in S_l} p_i$. As $t(S_h) + t(S_l) = \Sigma$, we have

$$t(S_h) - \frac{1+\alpha}{2}L = t(S_h) - \frac{1+\alpha}{2} \cdot \frac{t(S_h) + t(S_l)}{2} = \frac{3-\alpha}{4} t(S_h) - \frac{1+\alpha}{2} \cdot \frac{t(S_l)}{2}. \quad (1)$$

As $\alpha < 2$, we have $\frac{3-\alpha}{4} > 0$. Combined with $t(S_h) \leq \frac{n}{2}\alpha$ and $t(S_l) \geq \frac{n}{2}$, we get

$$t(S_h) - \frac{1+\alpha}{2}L \leq \frac{3-\alpha}{4} \cdot \frac{n}{2} \cdot \alpha - \frac{1+\alpha}{2} \cdot \frac{n}{4} = -\frac{n}{8} \cdot (\alpha - 1)^2 \leq 0, \quad (2)$$

which means that the total processing time of arbitrary $\frac{n}{2}$ jobs in σ is at most $\frac{1+\alpha}{2}L$. The proof is completed. □

Straightforwardly, we have the following corollary.

Corollary 1 *Suppose that* $1 \leq \alpha < 2$. *For an arbitrary job sequence* σ *which contains* n *jobs, the total processing time of arbitrary* n' *jobs in* σ *is at most* $\frac{1+\alpha}{2}L$ *when* $n' \leq \lfloor \frac{n}{2} \rfloor$.

To prove that algorithm M is optimal for arbitrary α, we prove algorithm M is $\frac{1+\alpha}{2}$-competitive when $1 \leq \alpha < 2$ and is $\frac{3}{2}$-competitive when $\alpha \geq 2$, respectively.

Lemma 2 *Algorithm* M *is* $\frac{1+\alpha}{2}$*-competitive when* $1 \leq \alpha < 2$.

Proof Suppose that the lemma is false, then there must exist at least one instance I which makes $\frac{C_M}{C_{opt}} > \frac{1+\alpha}{2}$. Let n be the number of the jobs in I. We distinguish the following two cases according to the value of n.

Case 1: n is an even number.

In this case, we have two subcases. The first subcase is that no jobs with $GoS = 2$ are assigned to S_1. If no jobs with $GoS = 2$ are assigned to S_1, according to the rules of algorithm M, we have $t(S_2) \leq \frac{1+\alpha}{2}L \leq \frac{1+\alpha}{2}C_{opt}$. And $C_{opt} \geq t(S_1)$ holds since all the jobs assigned to S_1 are with $GoS = 1$. Hence, in this case, $C_M = \max\{t(S_1), t(S_2)\} \leq \frac{1+\alpha}{2}C_{opt}$.

The other subcase is that there is at least one job with $GoS = 2$ assigned to S_1. Let J_i denote the first job with $GoS = 2$ assigned to S_1 by algorithm M, then we have $t(S_2^{i-1}) + p_i > \beta L$. Based on Lemma 1, algorithm M assigns at least $\frac{n}{2}$ jobs of I to S_2 before scheduling job J_i, otherwise $t(S_2^{i-1}) + p_i \leq \beta L$ holds since the total processing time of arbitrary $\frac{n}{2}$ jobs is at most $\frac{1+\alpha}{2}L$. As the number of the jobs in I is n and more than half of the jobs were assigned to S_2 before scheduling job J_i, the number of the jobs that didn't assigned to S_2^{i-1} is also at most $\frac{n}{2}$, which implies that $\Sigma - t(S_2^{i-1}) \leq \beta L$. So algorithm M will stop at Step 4.2. Again, we get $C_M = \max\{t(S_1), t(S_2)\} \leq \beta L \leq \frac{1+\alpha}{2}C_{opt}$.

Case 2: n is an odd number.

Divide I into two subsets I_1 and I_2 where I_1 contains $\frac{n+1}{2}$ jobs and I_2 contains $\frac{n-1}{2}$ jobs. The processing time of any job in I_1 is not greater than the processing time of any job in I_2. Let $t(I_1)$ and $t(I_2)$ denote the total processing time of the jobs in I_1 and I_2, respectively. Note that $t(I_1) + t(I_2) = \Sigma$. As the processing times are bounded in the interval $[1, \alpha]$, we have $t(I_1) \geq \frac{n+1}{2}$ and $t(I_2) \leq \frac{n-1}{2}\alpha$. According to the definition of I_1, I_1 contains $\frac{n+1}{2}$ jobs which have the most shortest processing time. Since the optimal algorithm must assign at least $\frac{n+1}{2}$ jobs to one of the two machines, $C_{opt} \geq t(I_1)$ holds.

If algorithm M stops at Step 4.2 or Step 5, we can directly get that $C_M = \max\{t(S_1), t(S_2)\} \leq \frac{1+\alpha}{2}C_{opt}$. Therefore, it stops at Step 4.3 or Step 4.4.

Suppose that algorithm M stops at Step 4.3, and J_i is the job which makes $t(S_2^{i-1}) + p_i > \frac{1+\alpha}{2}L$, $\Sigma - t(S_2^{i-1}) > \frac{1+\alpha}{2}L$ and $\Sigma - t(S_2^{i-1}) - p_i < t(S_2^{i-1})$ hold. In this case, we have $t(S_2) = t(S_2^{i-1})$ and $t(S_1) = \Sigma - t(S_2^{i-1})$. If $C_M = \max\{t(S_1), t(S_2)\} = t(S_2)$, we get $C_M \leq \frac{1+\alpha}{2}L \leq \frac{1+\alpha}{2}C_{opt}$ since $t(S_2) \leq \frac{1+\alpha}{2}L$. Otherwise, $C_M = \max\{t(S_1), t(S_2)\} = t(S_1) > \frac{1+\alpha}{2}C_{opt}$. As $\Sigma - t(S_2) - p_i < t(S_2)$ and $t(S_1) = \Sigma - t(S_2)$, we get $t(S_2) + p_i > t(S_1)$, which means $t(S_2) + p_i > \frac{1+\alpha}{2}C_{opt}$. Then we have $t(S_1) + t(S_2) + p_i > (1+\alpha)C_{opt}$. As $t(S_1) + t(S_2) + p_i = t(I_1) + t(I_2) + p_i$ and $(1+\alpha)C_{opt} \geq (1+\alpha)t(I_1)$, we have

$$t(S_1) + t(S_2) + p_i = t(I_1) + t(I_2) + p_i > (1+\alpha)t(I_1) \tag{3}$$

which means

$$t(I_2) + p_i > \alpha t(I_1) \geq \frac{n+1}{2}\alpha. \tag{4}$$

Since $t(I_2) \leq \frac{n-1}{2}\alpha$ and $p_i \leq \alpha$, we have $t(I_2) + p_i \leq \frac{n+1}{2}\alpha$ which contradicts with $t(I_2) + p_i > \frac{n+1}{2}\alpha$.

If algorithm M stops at Step 4.4, then algorithm M assigns at least $\frac{n+1}{2}$ jobs to S_2, and assigns at most $\frac{n-1}{2} \leq \lfloor \frac{n}{2} \rfloor$ jobs to S_1. Based on Corollary 1, we have $t(S_1) \leq \frac{1+\alpha}{2}L \leq \frac{1+\alpha}{2}C_{opt}$. Hence, $C_M = t(S_2) = t(S_2^{i-1}) + p_i > \frac{1+\alpha}{2}C_{opt}$. In this case, according to algorithm M, we have $t(S_2^{i-1}) \leq \Sigma - t(S_2^{i-1}) - p_i = t(S_1)$, which leads to $t(S_1) + p_i \geq t(S_2) > \frac{1+\alpha}{2}C_{opt}$. Therefore, we have $t(S_1) + t(S_2) + p_i > (1+\alpha)C_{opt}$. Again, this leads to $t(I_2) + p_i \leq \frac{n+1}{2}\alpha$ which contradicts with $t(I_2) + p_i > \frac{n+1}{2}\alpha$.

Hence, we know that such an example which makes $\frac{C_M}{C_{opt}} > \frac{1+\alpha}{2}$ does not exist. The proof is completed. □

Lemma 3 *Algorithm M is $\frac{3}{2}$-competitive when $\alpha \geq 2$.*

Proof Suppose that this lemma is false, then there must exist at least one instance I which makes $\frac{C_M}{C_{opt}} > \frac{3}{2}$.

If algorithm M stops at Step 4.2 or Step 5, according to the algorithm, we have $C_M = \max\{t(S_1), t(S_2)\} \leq \frac{3}{2}C_{opt}$ directly. Therefore, it must stop at Step 4.3 or Step 4.4.

Suppose that algorithm M stops at Step 4.3, and J_i is the job which makes $t(S_2^{i-1}) + p_i > \frac{3}{2}L$, $\Sigma - t(S_2^{i-1}) > \frac{3}{2}L$ and $\Sigma - t(S_2^{i-1}) - p_i < t(S_2^{i-1})$ hold. In this case, we also have $t(S_2) = t(S_2^{i-1})$ and $t(S_1) = \Sigma - t(S_2^{i-1})$. According to the rules of algorithm M, if $C_M = \max\{t(S_1), t(S_2)\} = t(S_2)$, we have $C_M = t(S_2) \leq \frac{3}{2}L \leq \frac{3}{2}C_{opt}$. Therefore, $C_M = \max\{t(S_1), t(S_2)\} = t(S_1) > \frac{3}{2}C_{opt}$. As $\Sigma - t(S_2) - p_i < t(S_2)$ and $t(S_1) = \Sigma - t(S_2)$, we have $t(S_2) + p_i > t(S_1) > \frac{3}{2}C_{opt}$. Then we have $t(S_1) + t(S_2) + p_i > 3C_{opt} \geq 3L$. Combined with $t(S_1) + t(S_2) = \Sigma = 2L$, we have $p_i > L$ and $t(S_1) + t(S_2) - p_i < \Sigma - L = L$. Therefore, we get $\min\{t(S_1) - p_i, t(S_2)\} < \frac{1}{2}L$, otherwise $t(S_1) - p_i + t(S_2) \geq \frac{1}{2}L + \frac{1}{2}L = L$. Since J_i is assigned to S_1, we have

$$\min\{t(S_1) - p_i, t(S_2)\} = t(S_1) - p_i < \frac{1}{2}L < \frac{1}{2}p_i, \tag{5}$$

which leads to $t(S_1) < \frac{3}{2}p_i$. Combined with $C_{opt} \geq p_i$, we have $t(S_1) < \frac{3}{2}C_{opt}$ which contradicts with $t(S_1) > \frac{3}{2}C_{opt}$.

Suppose that algorithm M stops at step 4.4. If there are no jobs with $GoS = 2$ assigned to S_1, then we have $C_{opt} \geq t(S_1)$. Otherwise, we have $t(S_1) \leq \frac{3}{2}L \leq \frac{3}{2}C_{opt}$. Then $C_M = \max\{t(S_1), t(S_2)\} = t(S_2) > \frac{3}{2}C_{opt}$. We can get the same contradiction in the same way just as algorithm M stops at step 4.3.

Hence, we know that such an example which makes $\frac{C_M}{C_{opt}} > \frac{3}{2}$ does not exist. The proof is completed. □

Note that we prove Lemmas 1–3 with arbitrary Σ. Therefore, based on Lemmas 2 and 3, we obtain the following theorem.

Theorem 1 *Algorithm M is optimal for arbitrary α and Σ.*

Acknowledgments This work is supported by National Natural Science Foundation of China under Grants 71071123, 60921003 and 71371129 and Program for Changjiang Scholars and Innovative Research Team in University under Grant IRT1173.

References

1. Chen, X., Xu, Z., Dósa, G., Han, X., Jiang, H.: Semi-online hierarchical scheduling problems with buffer or rearrangements. Inf. Process. Lett. **113**, 127–131 (2013)
2. Hwang, H., Chang, S., Lee, K.: Parallel machine scheduling under a grade of service provision. Comput. Oper. Res. **31**, 2055–2061 (2004)

3. Jiang, Y., He, Y., Tang, C.: Optimal online algorithms for scheduling on two identical machines under a grade of service. J. Zhejiang Univ. Sci. A **7**, 309–314 (2006)
4. Ji, M., Cheng, T.C.E.: An FPTAS for parallel-machine scheduling under a grade of service provision to minimize makespan. Inf. Process. Lett. **108**, 171–174 (2008)
5. Liu, M., Chu, C., Xu, Y., Zheng, F.: Semi-online scheduling on 2 machines under a grade of service provision with bounded processing times. J. Combin. Optim. **21**, 138–149 (2011)
6. Luo, T., Xu, Y., Luo, L., He, C.: Semi-online scheduling with two GoS levels and unit processing time. Theor. Comput. Sci. **521**, 62–72 (2014)
7. Park, J., Chang, S.K., Lee, K.: Online and semi-online scheduling of two machines under a grade of service provision. Oper. Res. Lett. **34**, 692–696 (2006)
8. Woeginger, G.J.: A comment on parallel-machine scheduling under a grade of service provision to minimize makespan. Inf. Process. Lett. **109**, 341–342 (2009)
9. Wu, Y., Ji, M., Yang, Q.: Optimal semi-online scheduling algorithms on two parallel identical machines under a grade of service provision. Int. J. Prod. Econ. **135**, 367–371 (2012)
10. Zhang, A., Jiang, Y., Fan, L., Hu, J.: Optimal online algorithms on two hierarchical machines with tighly-grouped processing times. J. Combin. Optim. (2013). doi:10.1007/s10878-013-9627-7