Research Article

Semi-Online Scheduling on Two Machines with GoS Levels and Partial Information of Processing Time

Taibo Luo and Yinfeng Xu

1 Business School, Sichuan University, Chengdu 610065, China
2 State Key Lab for Manufacturing Systems Engineering, Xi’an 710049, China

Correspondence should be addressed to Yinfeng Xu; yfxu@scu.edu.cn

Received 15 November 2013; Accepted 19 December 2013; Published 11 February 2014

Academic Editors: J. G. Barbosa and D. Oron

Copyright © 2014 T. Luo and Y. Xu. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

This paper investigates semi-online scheduling problems on two parallel machines under a grade of service (GoS) provision subject to minimizing the makespan. We consider three different semi-online versions with knowing the total processing time of the jobs with higher GoS level, knowing the total processing time of the jobs with lower GoS level, or knowing both in advance. Respectively, for the three semi-online versions, we develop algorithms with competitive ratios of 3/2, 20/13, and 4/3 which are shown to be optimal.

1. Introduction

Scheduling problem under a grade of service provision was first proposed by Hwang et al. [1]. The jobs should be assigned irrevocably to machines as soon as they arrive. Each job and machine are labeled with the GoS levels. A job can be processed by a particular machine if and only if the GoS level of the job is not less than that of the machine. In practice, the scheduling with GoS eligibility constraints is widely used. For more details, please refer to [1–3].

For the offline scheduling on $m$ machines under a grade of service provision, Hwang et al. [1] presented an LQ-LPT algorithm which has a tight bound of $5/4$ for two machines and $2 - 1/(m-1)$ for $m$ ($m \geq 3$) machines. Ji and Cheng [4] gave an FPTAS for this problem. However, Woeginger [5] gave two simpler FPTASs for the same problem. For the online version, Jiang [6] proved a lower bound of 2 for the case with two levels of GoS and presented an online algorithm with a competitive ratio of 2.52. Zhang et al. [7] improved the result and showed an algorithm with a competitive ratio of $1 + (m^2 - m)/(m^2 - sm + s^2) \leq 7/3$.

For $m = 2$, Park et al. [8] presented an optimal algorithm with a competitive ratio of 5/3. However, there are many papers focusing on the semi-online scheduling on two machines under GoS [8–13]. As the total processing time of all jobs is known, Wu et al. [12] presented an optimal algorithm with a competitive ratio of $4 \sqrt{2} + 1)/2$. In the same paper, Wu et al. [12] gave an optimal algorithm with a competitive ratio of 3/2 when the optimal value of the instance is known. For the processing times bounded in an interval, Liu et al. [9] gave a competitive algorithm under some conditions and Zhang et al. [13] improved the result and gave an optimal algorithm for different intervals. In this paper, we focus on the semi-online scheduling problem where only partial information of the total processing time is known. The semi-online versions concerned in our paper are listed as follows.

- $s_{\text{sum\cdot higher}}$: the total processing time of all jobs with higher GoS level is known in advance.
- $s_{\text{sum\cdot lower}}$: the total processing time of all jobs with lower GoS level is known in advance.

We use $P_2 \cdot \text{GoS} \mid s \mid C_{\text{max}}$ to denote the semi-online problem with information $s$, where $s \in \{s_{\text{sum\cdot higher}}, s_{\text{sum\cdot lower}}\}$. Moreover, we use $P_2 \cdot \text{GoS} \mid s_1 \& s_2 \mid C_{\text{max}}$ to denote the semi-online problem where both information $s_1$ and information $s_2$ are available in advance.

Our results indicate that competitive ratios of three algorithms are better than 5/3 of the online version [8]. When knowing $s_{\text{sum\cdot higher}}$ in advance, the competitive ratio is
the same as the optimal algorithm [8] presented for the semi-online problem where the total processing time of all jobs is known. For designing algorithm, the results indicate that knowing sum · higher & sum · lower is more useful than sum · higher or sum · lower. Moreover, knowing sum · higher is more useful than sum · lower. However, knowing sum · lower is better than knowing the largest processing time of all jobs [12].

The rest of this paper is organized as follows: in Section 2, we give some basic definitions. In Sections 3–5, we prove lower bounds and present algorithms for the three semi-online problems, respectively.

2. Basic 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 arrival of a new job occurs only after the current job is scheduled. We denote by the set \( J = 1, \ldots, n \) the set of all job indices arranged in the order of arrival. The \( j \)th arriving job is referred to as job \( J_j \). We denote each job by \( J_j = (p_j, g_j) \). The GoS assigned to job \( J_j \) is denoted by \( g_j \), which is 1 if the job must be processed only by the first machine and 2 if it may be processed by either of the two; \( p_j \) is the processing time of job \( J_j \); \( p_j \) and \( g_j \) are not known until the arrival of job \( J_j \). Let \( T_0 = \sum_{j=1}^{n} p_j \) and \( T_2 = \sum_{j=2}^{n} p_j \).

The scheduled can be seen as the partition of \( J \) into two subsets, denoted by \((S_1, S_2)\), where \( S_1 \) and \( S_2 \) contain job indices assigned to the first and second machine, respectively. Let \( t(S) = \sum_{j \in S} p_j \) for an arbitrary subset \( S \) of \( J \). Then the maximum of \( t(S_1) \) and \( t(S_2) \), denoted by \( \max(t(S_1), t(S_2)) \), is the makespan of the scheduled \((S_1, S_2)\). The problem is to minimize \( \max(t(S_1), t(S_2)) \).

The minimum makespan obtained by an optimal offline algorithm is denoted by \( C_{\text{opt}} \). The makespan generated by algorithm \( A \) is denoted by \( C_A \). The competitive ratio of \( A \) is defined to be the maximum \( C_A / C_{\text{opt}} \).

3. Optimal Algorithm for \( P2 \cdot \text{GoS} \mid \text{sum} \cdot \text{higher} \mid C_{\text{max}} \)

In this section, we prove a lower bound of competitive ratio and present an optimal algorithm for the semi-online version as \( T_1 \) is known in advance.

3.1. Lower Bound of Competitive Ratio

Theorem 1. Any semi-online algorithm for \( P2 \cdot \text{GoS} \mid \text{sum} \cdot \text{higher} \mid C_{\text{max}} \) has a competitive ratio of at least \( 3/2 \).

Proof. The theorem will be proved by adversary method. Let \( T_1 = 1 \) be known in advance. The first job is \( J_1 = (1, 1) \). \( J_1 \) must be scheduled on the first machine. Then, we generate job \( J_2 = (1, 2) \). If job \( J_2 = (1, 2) \) is scheduled on the first machine, and there is no job arriving after, we have \( C_A / C_{\text{opt}} = 2 > 3/2 \). Otherwise, job \( J_2 \) is scheduled on the second machine, and we generate job \( J_3 = (2, 2) \). No matter which machine that job \( J_3 \) is scheduled on, we have \( C_A = 3 \). Since the optimal algorithm will scheduled job \( J_1 \) and job \( J_2 \) on the first machine and scheduled job \( J_3 \) on the second machine. Hence, \( C_A / C_{\text{opt}} = 3/2 \). The proof is completed.

3.2. Optimal Semi-Online Algorithm GoS-TH. Since we know \( T_1 \) in advance and all the jobs with \( g_j = 1 \) must be scheduled on the first machine, we can regard them as one job; that is, \( J_0 = (T_1, 1) \). We scheduled job \( J_0 \) on the first machine at first and do not need to care about the job with \( g_j = 1 \) later.

At the arrival of each job, \( H \) is updated to become a half of the total processing time of the jobs which include the jobs with GoS = 2 and job \( J_0 \); \( P \) is updated to become the maximum processing time. We define \( P^0, S_1^1, S_2^1, \text{and} H^1 \) to be \( P, S_1, S_2, \text{and} H \) after we scheduled job \( J_j \). Then, clearly the optimum makespan \( C_{\text{opt}} \geq L = \max(T, P) \). Combined with the online algorithm presented by Park et al. [8], we propose Algorithm GoS-TH.

Algorithm GoS-TH

1. Scheduled \( J_0 \) to the first machine and \( t(S_1^1) = T_1 \).
2. Let \( p_j = 0 \) for all the jobs with \( g_j = 1 \).
3. Suppose that the incoming job is \( J_j \) and \( g_j = 1 \); assign job \( J_j \) to the first machine.
4. Suppose the incoming job is \( J_j \) and \( g_j = 2 \). \( H_j = H_{j-1} + p_j / 2, P_j = \max(P_{j-1}, p_j) \), and \( H_j = \max(T, P_j) \).
   (4.1) If \( t(S_1^1) + p_j = (3/2)L_j \), scheduled it to the second machine.
   (4.2) If \( t(S_2^1) + p_j > (3/2)L_j \), scheduled it to the first machine.

Theorem 2. The competitive ratio of Algorithm GoS-TH is \( 3/2 \) for \( P2 \cdot \text{GoS} \mid \text{sum} \cdot \text{higher} \mid C_{\text{max}} \).

Proof. Suppose that Theorem 2 is false. There must exist an instance with the least number of jobs to make \( C_{\text{GoS-TH}} > (3/2)C_{\text{opt}} \). The makespan is not determined until the arrival of job \( J_n \). Therefore,

\[
C_{\text{GoS-TH}} = \max \{ t(S_1^n), t(S_2^n) \} > 3/2C_{\text{opt}},
\]

\[
\max \{ t(S_1^{n-1}), t(S_2^{n-1}) \} \leq 3/2C_{\text{opt}}.
\]

If \( g_n = 1 \), based on Algorithm GoS-TH, we have \( t(S_1^n) = t(S_1^{n-1}) \) and \( t(S_2^n) = t(S_2^{n-1}) \). Due to the fact that \( \max(t(S_1^{n-1}), t(S_2^{n-1})) \leq (3/2)C_{\text{opt}} \), we just need to prove Theorem 2 is true when \( g_n = 2 \).

If \( g_n = 2 \), when it is scheduled on the second machine, we have \( C_{\text{GoS-TH}} = t(S_2^n) = t(S_2^{n-1}) + p_n \leq (3/2)C_{\text{opt}} \). This is contradicting with inequality (1). Otherwise, job \( J_n \) is scheduled on the first machine, which leads to \( t(S_2^{n-1}) + p_n > (3/2)L_n \geq (3/2)H_n \); combined with \( (t(S_1^{n-1}) + p_n + t(S_2^{n-1}))/2 = H_n \), we get that \( t(S_1^{n-1}) < (1/2)L_n \leq (1/2)C_{\text{opt}} \). Since \( p_n \leq C_{\text{opt}} \), we have \( t(S_1^{n-1}) + p_n < (3/2)C_{\text{opt}} \), which also contradicted with inequality (1). The proof is completed.
4. Optimal Algorithm for

\[ P2 \cdot GoS \mid \text{sum} \cdot \text{lower} \mid C_{\text{max}} \]

In this section, we prove a lower bound of competitive ratio and present an optimal algorithm for the semi-online version as \( T_2 \) is known in advance.

4.1. Lower Bound of Competitive Ratio

**Theorem 3.** Any semi-online algorithm for \( P2 \cdot GoS \mid \text{sum} \cdot \text{lower} \mid C_{\text{max}} \) has a competitive ratio of at least 20/13.

**Proof.** We will construct a job sequence with \( T_2 = 26 \) to make an arbitrary algorithm \( A \) behave poorly. We begin with jobs \( J_1 = (1,2) \) and \( J_2 = (1,2) \). We discuss the following three cases.

**Case 1** \((J_1 \text{ and } J_2 \text{ are scheduled on the first machine}).\) We continue to generate jobs \( J_3 = (12,2) \) and \( J_4 = (12,2) \). If job \( J_3 \) and job \( J_4 \) are scheduled on the second machine, we have \( t(S_3) = 24 \); thus, \( C_A/C_{\text{opt}} = 24/14 > 20/13 \). Otherwise, if job \( J_3 \) or job \( J_4 \) is scheduled on the first machine or both of them are scheduled on the first machine, we further generate job \( J_5 = (26,1) \). We will have \( t(S_4) \geq 40 \). Since optimal algorithm will scheduled jobs \( J_1, J_2, J_3, \) and \( J_4 \) on the second machine and scheduled job \( J_5 \) on the first machine, we have \( C_A/C_{\text{opt}} \geq 20/13 \).

**Case 2** \((J_1 \text{ or } J_2 \text{ is scheduled on the first machine}).\) We continue to generate job \( J_3 = (2,2) \). Then we discuss the following two subcases.

**Subcase 2.1** \((J_3 \text{ is scheduled on the first machine}).\) We continue to generate jobs \( J_4 = (11,2) \) and \( J_5 = (11,2) \). If job \( J_4 \) and job \( J_5 \) are scheduled on the second machine, we have \( t(S_4) = 23 \) and \( t(S_5) = 2 \), which lead to \( C_A/C_{\text{opt}} = 23/13 < 20/13 \). Otherwise, if job \( J_4 \) or job \( J_5 \) is scheduled on the first machine or both of them are scheduled on the first machine, we further generate job \( J_6 = (26,1) \) which leads to \( C_A/C_{\text{opt}} \geq 20/13 \).

**Subcase 2.2** \((J_3 \text{ is scheduled on the second machine}).\) We generate job \( J_4 = (5,2) \). If job \( J_4 \) is scheduled on the second machine, we generate job \( J_5 = (13,2) \). If job \( J_5 \) is scheduled on the second machine, we have \( t(S_5) = 21 \). Since the optimal algorithm will scheduled jobs \( J_1, J_2, J_3, \) and \( J_4 \) on the first machine and scheduled job \( J_5 \) on the second machine, we have \( C_A/C_{\text{opt}} = 21/13 < 20/13 \). If job \( J_5 \) is scheduled on the first machine, we generate job \( J_6 = (26,1) \) which leads to \( C_A/C_{\text{opt}} \geq 20/13 \).

Therefore, job \( J_5 \) is scheduled on the first machine. Then we generate jobs \( J_6 = (8,2) \) and \( J_7 = (8,2) \). If job \( J_6 \) or job \( J_7 \) is scheduled on the first machine or both of them are scheduled on the first machine, we further generate job \( J_8 = (26,1) \). We will have \( t(S_6) \geq 40 \) and \( C_{\text{opt}} = 26 \); hence, \( C_A/C_{\text{opt}} \geq 20/13 \).

**Case 3** \((J_1 \text{ and } J_2 \text{ are scheduled on the second machine}).\) We continue to generate job \( J_3 = (2,2) \). Then we discuss the following two subcases.

**Subcase 3.1** \((J_3 \text{ is scheduled on the second machine}).\) We generate jobs \( J_4 = (3,2) \) and \( J_5 = (3,2) \). If job \( J_4 \) or job \( J_5 \) is scheduled on the second machine or both of them are scheduled on the second machine, we further generate job \( J_6 = (1,2) \). If job \( J_6 \) is scheduled on the second machine, we generate job \( J_7 = (14,2) \). If job \( J_7 \) is scheduled on the second machine, we have \( t(S_7) = 22 \). Since the optimal algorithm will scheduled jobs \( J_1, J_2, J_3, J_4, J_5, J_6, \) and \( J_7 \) on the first machine and scheduled job \( J_8 \) on the second machine, we have \( C_A/C_{\text{opt}} \geq 22/14 = 20/13 \). Otherwise, job \( J_7 \) is scheduled on the first machine, and then we generate job \( J_8 = (26,1) \) and have \( t(S_7) \geq 40 \) and \( C_{\text{opt}} = 26 \); hence, \( C_A/C_{\text{opt}} > 20/13 \). Otherwise, job \( J_6 \) is scheduled on the first machine, and then we generate job \( J_6 = (26,1) \). We have \( C_A/C_{\text{opt}} > 20/13 \).

If job \( J_4 \) and job \( J_5 \) are scheduled on the first machine, we generate jobs \( J_6 = (8,2) \) and \( J_7 = (8,2) \). If job \( J_6 \) or job \( J_7 \) is scheduled on the first machine or both of them are scheduled on the first machine, we further generate job \( J_8 = (26,1) \). We have \( C_A/C_{\text{opt}} > 20/13 \). Otherwise, if job \( J_6 \) and job \( J_7 \) are scheduled on the second machine, we have \( t(S_7) \geq 20 \) and \( C_{\text{opt}} = 13 \), which lead to \( C_A/C_{\text{opt}} = 20/13 \).

**Subcase 3.2** \((J_3 \text{ is scheduled on the first machine}).\) We generate job \( J_4 = (2,2) \). If job \( J_4 \) is scheduled on the first machine, we generate jobs \( J_5 = (10,2) \) and \( J_6 = (10,2) \). If job \( J_5 \) and job \( J_6 \) are scheduled on the second machine, we have \( t(S_5) = 22 \) and \( C_{\text{opt}} = 13 \); thus, \( C_A/C_{\text{opt}} = 22/13 > 20/13 \). Otherwise, if job \( J_5 \) or job \( J_6 \) is scheduled on the first machine or both of them are scheduled on the first machine, we further generate job \( J_7 = (26,1) \) and have \( t(S_7) \geq 40 \); thus, \( C_A/C_{\text{opt}} > 20/13 \).

Otherwise, if job \( J_4 \) is scheduled on the second machine, we generate job \( J_5 = (4,2) \). If job \( J_5 \) is scheduled on the second machine, we generate job \( J_6 = (13,2) \). If job \( J_6 \) is scheduled on the second machine, we have \( t(S_6) = 21 \) and \( C_{\text{opt}} = 13 \); hence, \( C_A/C_{\text{opt}} > 20/13 \). If job \( J_6 \) is scheduled on the first machine, we generate job \( J_7 = (26,1) \) and have \( t(S_7) \geq 40 \) and \( C_{\text{opt}} = 26 \); hence, \( C_A/C_{\text{opt}} > 20/13 \).

The proof is completed.

4.2. Optimal Semi-Online Algorithm GoS-TL. In this subsection, we design an optimal algorithm with a competitive ratio of 20/13. Let \( x_1 \) and \( x_2 \) be the jobs with \( g_j = 2 \) assigned
to the first and second machine, respectively, where \( t(x_1) + t(x_2) = T_2 \). We define \( x_1 \) and \( x_2 \) to be \( x_1 \) and \( x_2 \) after we scheduled job \( J_1 \). Then, we propose Algorithm GoS-TL.

**Algorithm GoS-TL**

1. Suppose that the incoming job is \( J_1 \) and \( g_1 = 1 \); assign job \( J_1 \) to the first machine.
2. Suppose that the incoming job is \( J_1 \) and \( g_2 = 2 \).

   2.1. *(Stopping criterion 1)*. If \((3/13)T_2 \leq t(x_1) + p_j \leq (7/13)T_2\), scheduled job \( J_1 \) and all the remaining jobs with GoS = 1 to the first machine, and then scheduled all the remaining jobs with GoS = 2 to the second machine. Stop.

   2.2. *(Stopping criterion 2)*. If \((6/13)T_2 \leq t(x_1) + p_j \leq (10/13)T_2\), scheduled job \( J_1 \) to the second machine and scheduled all the remaining jobs to the first machine. Stop.

   2.3. *(Stopping criterion 3)*. If \( t(x_2) > 10/13 \), scheduled job \( J_1 \) to the second machine. Stop.

   2.4. If \( t(x_2) \leq (7/26)T_2 \) and \( (7/13)T_2 \leq t(x_2) + p_j \leq (6/13)T_2\), scheduled job \( J_1 \) to the first machine. Continue.

2.5. Otherwise, scheduled job \( J_1 \) to the second machine. Continue.

**Lemma 4.** If \((6/13)T_2 \leq t(x_2) \leq (10/13)T_2\), then \( C_{\text{GoS-TL}} \leq (20/13)C_{\text{opt}}\).

**Proof.** Since \( C_{\text{opt}} \geq (1/2)T_2 \), we have \( t(S_2) = t(x_2) \leq (10/13)T_2 \leq (20/13)C_{\text{opt}}\). Then, we only need to prove that \( t(S_1) \leq (20/13)C_{\text{opt}}\). We discuss it by the following two cases.

**Case 1** \((T_1 \geq T_2)\). In this case, we have \( t(S_1) = t(x_1) \) and \( T_1 \geq (T_1 + T_2)/2\). Since \((6/13)T_2 \leq t(x_2) \leq (10/13)T_2\), we have \( t(x_1) \leq (7/13)T_2\). Combined with \( C_{\text{opt}} \geq T_1\), we have

\[
\frac{C_{\text{GoS-TL}}}{C_{\text{opt}}} \leq \frac{(7/13)T_2 + T_1}{T_1} \leq \frac{(7/13)T_1 + T_1}{T_1} = \frac{20}{13}.
\]

**Case 2** \((T_1 < T_2)\). In this case, we have \( T_1 < (T_1 + T_2)/2\), which leads to \( C_{\text{opt}} \geq (T_1 + T_2)/2\). Then, we have

\[
\frac{C_{\text{GoS-TL}}}{C_{\text{opt}}} \leq \frac{(14/13)T_2 + 2T_1}{T_1 + T_2}.
\]

Since \((14/13)T_2 + 2T_1)/(T_1 + T_2)\) is increasing function of the variation of \( T_1\), thus, we have

\[
\frac{C_{\text{GoS-TL}}}{C_{\text{opt}}} \leq \frac{(14/13)T_2 + 2T_1}{T_1 + T_2} \leq \frac{(14/13)T_2 + 2T_2}{T_2 + T_2} = \frac{20}{13}.
\]

The proof is completed.

**Based on Lemma 4, we straightforwardly have Corollary 5.**

**Corollary 5.** *(1)* If \( t(x_2) \leq (7/13)T_2\), then \( t(S_1) \leq (20/13)C_{\text{opt}}\); *(2)* if \( t(x_2) > (10/13)T_2\), then \( t(S_2) \leq (20/13)C_{\text{opt}}\).

**Lemma 6.** \( t(x_1) \leq (7/13)T_2\).

**Proof.** Since Algorithm GoS-TL will only scheduled a job with \( g_1 = 2 \) to the first machine only when \( t(x_1) + p_j \leq (7/13)T_2\), the lemma can directly be got from the Algorithm GoS-TL. The proof is completed.

By using Corollary 5 and Lemma 6, we can obtain the following corollary.

**Corollary 7.** If \( t(x_2) \leq (10/13)T_2\), then \( C_{\text{GoS-TL}} \leq (20/13)C_{\text{opt}}\).

Based on Lemma 6 and Corollary 7, if we prove \( t(S_2) \leq (20/13)C_{\text{opt}}\) will hold when \( t(x_2) > (10/13)T_2\), then we can prove that Algorithm GoS-TL is \( (20/13)\)-competitive.

**Lemma 8.** If job \( J_1 \) is scheduled on the second machine by Algorithm GoS-TL where \( t(x_2) + p_j > (10/13)T_2\) and \( t(x_2) \leq (7/26)T_2\), then \( t(S_2) < (20/13)C_{\text{opt}}\).

**Proof.** If \( t(x_2) + p_j > (10/13)T_2\) and \( t(x_2) \leq (7/26)T_2\), we have \( p_j > (1/2)T_2\). Then we have \( C_{\text{opt}} \geq p_j > (1/2)T_2\). If job \( J_1 \) is scheduled on the second machine, Algorithm GoS-TL will scheduled the remaining jobs on the first machine. Thus,

\[
\frac{t(S_2)}{C_{\text{opt}}} \leq \frac{t(x_2) + p_j}{p_j} \leq \frac{(7/26)T_2 + (1/2)T_2}{(1/2)T_2} = \frac{20}{13}.
\]

The proof is completed.

**Lemma 9.** If \((6/13)T_2 \leq t(x_2) \leq (10/13)T_2\), then \( t(S_2) > (10/13)T_2\) will never happen.

**Proof.** If \((6/13)T_2 \leq t(x_2) \leq (10/13)T_2\), Algorithm GoS-TL will scheduled the remaining jobs on the first machine, so \( t(S_2) = t(x_2) \leq (10/13)T_2\). The proof is completed.

**Lemma 10.** If job \( J_1 \) is scheduled on the second machine by Algorithm GoS-TL where \((7/26)T_2 < t(x_2) < (6/13)T_2\) and \( t(x_2) > (10/13)T_2\), then \( t(S_2) = t(x_2) < (20/13)C_{\text{opt}}\).

**Proof.** Let \( J_w \) be the job that is scheduled on the second machine by Algorithm GoS-TL which satisfies \( t(x_2) \leq 7/13 \) and \((7/26)T_2 < t(x_2) < (6/13)T_2\). Since \( J_w \) is
scheduled on the second machine by Algorithm GoS-TL and $(7/26)T_2 < t(x_{w-1}^w) + p_w < (6/13)T_2$, we have $t(x_{w-1}^w) + p_w > (7/13)T_2$; otherwise, Algorithm GoS-TL will schedule job $J_w$ on the first machine. Moreover, $t(x_{w-1}^w) < (3/13)T_2$ must hold which leads to $p_w > (4/13)T_2$. This further implies that $t(x_{w-1}^w) < (2/13)T_2$ since $t(x_{w-1}^w) + p_w < (6/13)T_2$.

If $t(x_{w-1}^w) = 0$, then $p_w > (7/13)T_2$ which leads to $t(x_{w}^w) < (6/13)T_2$; this is contradicting the definition of job $J_w$. Therefore, we have $t(x_{w-1}^w) > 0$. Based on step (2.4) of Algorithm GoS-TL, the processing time of the job that is assigned to $x_{w-1}^w$ is larger than $(3/26)T_2$ since $t(x_{w-1}^w) < (2/13)T_2$. Combined with $t(x_{w-1}^w) < (3/13)T_2$, we know there is only one job in $x_{w-1}^w$, call it job $J_w$. Since job $J_w$, job $J_w$, and $t(x_{w-1}^w)$ need to satisfy

\[
p_v > \frac{7}{13}T_2,
\]

\[
p_v + t(x_{w-1}^w) > \frac{7}{26}T_2,
\]

\[
\frac{7}{26}T_2 < t(x_{w-1}^w) + p_w < \frac{6}{13}T_2,
\]

we can get $p_v > (9/52)T_2$, which implies that $t(S_2) < (43/52)T_2$.

Since there is no job with $j = 2$ that will scheduled on the first machine between job $J_1$ and $J_j$ by Algorithm GoS-TL, combined with job $J_j$ being scheduled on the second machine, we have $p_v + p_j > (7/13)T_2$. Since $p_j > (4/13)T_2$, $p_v$, $p_w$, and $p_j$ must satisfy

\[
p_v + p_j > \frac{7}{13}T_2,
\]

\[
p_v + p_j > \frac{7}{13}T_2,
\]

\[
p_w + p_j < \frac{9}{13}T_2,
\]

which implies that $C_{\text{opt}} > (7/13)T_2$, hence,

\[
\frac{t(S_2)}{C_{\text{opt}}} < \frac{(43/52)T_2}{(7/13)T_2} = \frac{20}{13}.
\]

The proof is completed.

Based on Lemma 4 to Lemma 10 and Corollary 5 to Corollary 7, we have the following theorem naturally.

**Theorem 11.** The competitive ratio of Algorithm GoS-TL is $20/13$ for $P2 \cdot \text{GoS} \mid \text{sum} \cdot \text{lower} \mid C_{\text{max}}$.

### 5. Optimal Algorithm for $P2 \cdot \text{GoS} \mid \text{sum} \cdot \text{high} & \text{sum} \cdot \text{lower} \mid C_{\text{max}}$

In this section, we show a lower bound of competitive ratio and present an optimal algorithm for the semi-online version as $T_1$ and $T_2$ are known in advance.

#### 5.1. Lower Bound of Competitive Ratio

**Theorem 12.** Any semi-online algorithm for $P2 \cdot \text{GoS} \mid \text{sum} \cdot \text{high} & \text{sum} \cdot \text{lower} \mid C_{\text{max}}$ has a competitive ratio of at least $4/3$.

**Proof.** The theorem will be proved by adversary method. Let $T_1 = 1/3$ and $T_2 = 5/3$ be known in advance. The first job is $J_1 = (1,3,1)$. Job $J_1$ must be scheduled on the first machine. Then job $J_2 = (1,3,2)$ arrives. If job $J_3$ is scheduled on the second machine, we further generate jobs $J_3 = (1,2)$ and $J_4 = (1,3,2)$. In this situation, we have $C_A \geq 4/3$ and $C_{\text{opt}} = 1$ since the optimal algorithm will scheduled jobs $J_1$, $J_2$, and $J_4$ on the first machine and scheduled job $J_3$ on the second machine. Thus, we have $C_A/C_{\text{opt}} \geq 4/3$. Otherwise, if job $J_3$ is scheduled on the first machine, then we further generate jobs $J_3 = (2,3,2)$ and $J_4 = (2,3,2)$. In this situation, we also have $C_A \geq 4/3$ and $C_{\text{opt}} = 1$ since the optimal algorithm will scheduled jobs $J_1$ and $J_3$ on the first machine and scheduled jobs $J_2$ and $J_3$ on the second machine. The proof is completed.

#### 5.2. Optimal Semi-Online Algorithm GoS-TB

In this subsection, we design an optimal algorithm with a competitive ratio of $4/3$. Since we know $T_1$ in advance and all the jobs with $g_j = 1$ must be scheduled on the first machine, therefore, we can regard them as one job, that is, $J_0 = (T_1,1)$. We scheduled job $J_0$ on the first machine at first and do not need to care about the job with $g_j = 1$ later. We present Algorithm GoS-TB as follows.

**Algorithm GoS-TB**

1. Scheduled $J_0$ to the first machine and $t(S_0) = T_1$.
2. Let $p_j = 0$ for all the jobs with $g_j = 1$.
3. Suppose that the incoming job is $J_j$ and $g_j = 1$; assign job $J_j$ to the first machine.
4. Suppose that the incoming job is $J_j$ and $g_j = 2$.

4.1. If $t(S_1) + p_j \leq (2/3)(T_1 + T_2)$, scheduled it to the first machine.

4.2. **(Stopping criterion)**. Suppose $t(S_1) + p_j > (2/3)(T_1 + T_2)$.

4.2.1. If $t(S_1^{j-1}) > (1/3)(T_1 + T_2)$, scheduled job $J_j$ and the remaining jobs with GoS = 2 on the second machine and scheduled the remaining jobs with GoS = 1 to the first machine. Stop.

4.2.2. If $t(S_1^{j-1}) \leq (1/3)(T_1 + T_2)$, scheduled job $J_j$ to the second machine, and scheduled the remaining jobs to the first machine. Stop.

**Theorem 13.** The competitive ratio of Algorithm GoS-TB is $4/3$ for $P2 \cdot \text{GoS} \mid \text{sum} \cdot \text{high} & \text{sum} \cdot \text{lower} \mid C_{\text{max}}$.

**Proof.** Since the job with $g_j = 1$ is scheduled at first and we do not need to care about them after that. We focus on the jobs with $g_j = 2$. 

Assume job \( J_j \) is the first job with \( g_j = 2 \) to make \( t(S_1) + p_j > (2/3)(T_1 + T_2) \). We prove it by the following two cases.

**Case 1** (\( t(S_1) > (1/3)(T_1 + T_2) \)). In this case, Algorithm GoS-TB schedules job \( J_j \) and the remaining jobs with GoS = 2 on the second machine. Therefore, \( t(S_1) \leq (2/3)(T_1 + T_2) \) and \( t(S_1) \leq T_1 + T_2 - t(S_1) < (2/3)(T_1 + T_2) \). Since \( C_{\text{opt}} \geq (1/2)(T_1 + T_2) \), we have \( C_{\text{GoS-TB}} = \max\{t(S_1), t(S_2)\} \leq (2/3)(T_1 + T_2) \leq (4/3)C_{\text{opt}} \).

**Case 2** (\( t(S_1) \leq (1/3)(T_1 + T_2) \)). In this case, we have \( p_j > (1/3)(T_1 + T_2) \). Algorithm GoS-TH will only scheduled job \( J_j \) on the second machine and scheduled the remaining jobs on the first machine. If \( p_j \geq (1/2)(T_1 + T_2) \), we have \( C_{\text{opt}} = p_j \) and \( t(S_1) \leq p_1 = t(S_2) \). Therefore, \( C_{\text{GoS-TH}} = C_{\text{opt}} \). Otherwise, \( (1/3)(T_1 + T_2) < p_j < (1/2)(T_1 + T_2) \), which leads to \( t(S_1) \leq T_1 + T_2 - t(S_1) < (2/3)(T_1 + T_2) \). Hence, we have \( C_{\text{GoS-TH}} = \max\{t(S_1), t(S_2)\} \leq (2/3)(T_1 + T_2) \leq (4/3)C_{\text{opt}} \). The proof is completed.

**Conflict of Interests**

The authors declare that there is no conflict of interests regarding the publication of this paper.

**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 IRT11173.

**References**

[1] H.-C. Hwang, S. Y. Chang, and K. Lee, “Parallel machine scheduling under a grade of service provision,” *Computers and Operations Research*, vol. 31, no. 12, pp. 2055–2061, 2004.

[2] A. Bar-Noy, A. Freund, and J. Naor, “On-line load balancing in a hierarchical server topology,” *SIAM Journal on Computing*, vol. 31, no. 2, pp. 527–549, 2001.

[3] J. Ou, J. Y.-T. Leung, and C.-L. Li, “Scheduling parallel machines with inclusive processing set restrictions,” *Naval Research Logistics*, vol. 55, no. 4, pp. 328–338, 2008.

[4] M. Ji and T. C. E. Cheng, “An FPTAS for parallel-machine scheduling under a grade of service provision to minimize makespan,” *Information Processing Letters*, vol. 108, no. 4, pp. 171–174, 2008.

[5] G. J. Woeginger, “A comment on parallel-machine scheduling under a grade of service provision to minimize makespan,” *Information Processing Letters*, vol. 109, no. 7, pp. 341–342, 2009.

[6] Y. Jiang, “Online scheduling on parallel machines with two GoS levels,” *Journal of Combinatorial Optimization*, vol. 16, no. 1, pp. 28–38, 2008.

[7] A. Zhang, Y. Jiang, and Z. Tan, “Online parallel machines scheduling with two hierarchies,” *Theoretical Computer Science*, vol. 410, no. 38–40, pp. 3597–3605, 2009.

[8] J. Park, S. Y. Chang, and K. Lee, “Online and semi-online scheduling of two machines under a grade of service provision,” *Operations Research Letters*, vol. 34, no. 6, pp. 692–696, 2006.

[9] M. Liu, C. Chu, Y. Xu, and F. Zheng, “Semi-online scheduling on 2 machines under a grade of service provision with bounded processing times,” *Journal of Combinatorial Optimization*, vol. 21, no. 1, pp. 138–149, 2011.

[10] M. Liu, Y. Xu, C. Chu, and F. Zheng, “Online scheduling on two uniform machines to minimize the makespan,” *Theoretical Computer Science*, vol. 410, no. 21-23, pp. 2099–2109, 2009.

[11] X. Lu and Z. Liu, “Online scheduling problems on two uniform machines under a grade of service provision,” *Theoretical Computer Science*, vol. 489–490, pp. 58–66, 2013.

[12] Y. Wu, M. Ji, and Q. Yang, “Optimal semi-online scheduling algorithms on two parallel identical machines under a grade of service provision,” *International Journal of Production Economics*, vol. 135, no. 1, pp. 367–371, 2012.

[13] A. Zhang, Y. Jiang, L. Fan, and J. Hu, “Optimal online algorithms on two hierarchical machines with tightly-grouped processing times,” *Journal of Combinatorial Optimization*, 2013.