Journal of the Operations Research Society of China (2022) 10:689–702
https://doi.org/10.1007/s40305-020-00325-6

Online Scheduling on Two Parallel Identical Machines Under a Grade of Service Provision

Shuang Cai[1] · **Ke Liu**[2,3]

Received: 1 February 2019 / Revised: 23 August 2019 / Accepted: 14 September 2020
Published online: 29 October 2020

Abstract
In this paper, we investigate online scheduling problems on two parallel identical machines under a grade of service provision with makespan as the objective function. The jobs arrive over time and each job can be scheduled only when it has already been arrived. We consider three different versions: (i) the two machines cannot be idle at the same time until all arrived jobs have been processed; (ii) further to the first problem, jobs are processed on a first-come, first-serviced basis; (iii) each job must be assigned to one of the two machines as soon as it arrives. Respectively for three online scheduling problems, optimal online algorithms are proposed with competitive ratio $(\sqrt{5}+1)/2$, $(\sqrt{5}+1)/2$ and $5/3$.

Keywords Online scheduling · Parallel machines · A grade of service provision · GoS

Mathematics Subject Classification 90B06 · 68M20

This research was supported by the National Natural Science Foundation of China (Nos. 11271356, and 71390334).

✉ Shuang Cai
caishuang@kingsoft.com

Ke Liu
kliu@amss.ac.cn

1 Data Department, Beijing Kingsoft Cloud Network Technology Co., Ltd., Beijing 100086, China

2 Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100190, China

3 University of Chinese Academy of Sciences, Beijing 100190, China

1 Introduction

Enterprise competition is becoming more and more fierce; lots of papers have studied the supply chain management [1]. Meanwhile, production and manufacturing become most important and complex. Parallel machine scheduling is one of the most common scheduling problems, which has been studied widely since 50 years ago [2]. Quite rich studies have been published for the offline scheduling problem on parallel machines, but rather few researches for the online scheduling problems. For the sake of simplicity, we use the 3-field notation of Graham et al. [3]. To evaluate the performance of an online algorithm, researchers use the competitive ratio of this algorithm, which is obtained by makespan of the online schedule divided by the minimum makespan of the schedule generated by the offline version of the problem, e.g. $c^A = \max_I(C_I^A/C_I^*)$, where c^A, C_I^A and C_I^* namely the competitive ratio of algorithm A, the makespan by algorithm A and the optimal makespan for instance I [2, 4–11].

Machine eligibility constraints are very common in the parallel machine scheduling and it is a common practice in any service industry to provide differentiated services to the customers based on their entitle privileges, i.e., jobs are assigned according to their grade of services (GoS) levels [4]. GoS means that customers with a higher level will receive better services by assigning a customer (job) to a server only when the GoS of the customer is no less than the GoS of the server. A few researches have been published for online and semi-online scheduling (see "Appendix A") on parallel machines under a grade of service provision. Lee et al. [2] provided a survey of online scheduling in parallel machine environments with machine eligibility constraints and the makespan as objective function.

When dealing with online scheduling problems on parallel machines, there are two basic online scheduling paradigms, namely online over list and online over time. For the version of online over list, lots of papers have been published [4–11]. Both Park et al. [4] and Jiang et al. [5] gave optimal algorithms with competitive ratio 5/3 and 3/2 for the online ($P2\big|\text{online}, M_j(\text{GoS})|C_{\max}$) and semi-online (for explanation, see "Appendix B") problems. Wu et al. [6] proposed two optimal algorithms for two different semi-online versions on two machines: The optimal offline value of the instance is known in advance or the largest processing time of all jobs is known in advance. Lu and Liu [12] studied the semi-online scheduling on two uniform machines under a GoS provision. The authors proposed optimal algorithms for three variants, where the optimal makespan, the total size of jobs, and the largest job size are known in advance, respectively.

A great amount of work has been done for the version of online over list, but very few researches have been published for the second version, despite the fact that it is more common and more realistic in our lives. For the online scheduling problem on parallel machines $Pm\big|\text{online}, r_j\,|C_{\max}$, Chen and Vestjens [13] proved that the algorithm of LPT has 3/2-competitive performance and the best ratio is 1.347; and Noga and Seiden [14] proposed an optimal algorithm for the two machine scheduling problem $P2\big|\text{online}, r_j\,|C_{\max}$ with competitive ratio 1.382. When preemption is allowed, Hong and Leung [15] provided an optimal algorithm for the problem $P2\big|\text{online}, r_j, pmtn|C_{\max}$ with a competitive ratio of 1. Lee et al. [16] proposed optimal algorithms for two problems concerning online scheduling of equal-length jobs on two machines subject to arbitrary eligibility constraints and GoS eligibility constraints, i.e., $P2\big|\text{online}, r_j, M_j, p_j = p|C_{\max}$ and $P2\big|\text{online}, r_j, M_j(\text{GoS}), p_j = p|C_{\max}$. Xu et al. [17] proposed an optimal algorithm for the online scheduling problem $P2\big|r_j, M_j(\text{GoS}), \text{online}|C_{\max}$ with competitive ratio 1.555 0. Li and Zhang [18] considered two scheduling problems $Q2\big|r_j, \text{online}|C_{\max}$ and $Pm\big|r_j, \text{online}|C_{\max}$ and proposed better algorithms for each of them. There are also some papers for parallel-batch machines scheduling. Recently, Yuan et al. [19] studied online scheduling on two uniform parallel-batch machines to minimize makespan. Cai and Liu [20] proposed heuristic algorithms for parallel machine scheduling problem $Pm\big|r_j, \text{online}, M_j(\text{GoS})|C_{\max}$.

In this paper, we consider three different versions for the online scheduling problem on two parallel identical machines under a grade of service provision (online over time), which have never been studied before. The first one is that we require the two machines cannot be idle at the same time until all arrived jobs have already completed. This constraint is more reasonable for real lives, e.g., the banks and the barber shop. It is because customer experience comes first and the persons with higher grade probably do not have a better experience than the ones with lower grade. Another important reason is that the average competitive ratio is usually lower. At the same time it usually adds the total completion time (the sum of two machines' completion times) when we let some customer wait while the two servers are both idle. The second problem is that we must assign jobs with the same GoS (1 or 2) to the two machines according to their arriving times, i.e., job i will be processed earlier than job j when job i arrives earlier and they have the same GoS, which is more equitable as it satisfies the method of FCFS (first-come-first-service). The third problem is that we must assign a job to its available machine as soon as it arrives. Respectively for the three problems, we propose algorithms which are proved to have optimal competitive ratios.

The rest of this paper is organized as follows: in Sect. 2, we give a detailed description of the first problem; then the best possible performance of any online algorithm is analyzed at the last of this section; we propose an online algorithm and it is proved with the best competitive ratio. In Sect. 3 (Sect. 4), the second (third) problem is studied with the same structure as Sect. 2. Finally, we give the conclusions and future research in Sect. 5.

2 Optimal Algorithm for the First Problem

In this section, firstly we describe the first problem; then, the lower bound of this problem is analyzed; at last, we propose an online algorithm, which is proved to be optimal.

2.1 Problem Description

The considered online scheduling problem can be described as: there are n different jobs J_i $(i = 1, \cdots, n)$, which have to be processed on one of the two parallel machines M_i $(i = 1, 2)$ without preemption. The GoS of machine M_i is i. Jobs are arrived over time with GoS (1 or 2). The two machines can be idle at the same time only when all the jobs arrived are completed.

The GoS of Job J_i is denoted as g_i, which is 1 if the job must be processed on the first machine M_1 or 2 if the job can be processed on both the two machines. The processing (arriving) time of job J_i is p_i (r_i). The constraint that the two machines can be idle at the same time only when all the jobs arrived are completed is denoted as no-delay1. The maximum complete time (makespan) is denoted as $C_{\max}$. Then, the considered problem in this paper can be described as $P2|\text{online},\ r_i,\ M(\text{GoS}),\ \text{no-delay1}|C_{\max}$. A job which is to be processed can be denoted by $(J,\ r,\ p,\ g)$. For example, (4, 3, 7, 1) represents that it is job J_4 with $r_4 = 3,\ p_4 = 7,\ g_4 = 1$.

To get the minimum makespan, we need to decide the assignment of each job after it arrives. The schedule can be seen as the partition of J into two sets, denoted by S_1 and S_2, where S_1 and S_2 contain jobs indices assigned to machine M_1 and M_2 respectively.

2.2 The Lower Bound of the First Problem

In this subsection, we give the lower bound of any online algorithm for the problem.

Theorem 1 *Any online algorithm for the first problem has a competitive ratio at least* $(\sqrt{5}+1)/2$.

Proof Note that $\lambda = (\sqrt{5}-1)/2$, C^* represents the optimal solution and C^A represents the solution of an online algorithm. At time $t = 0$, we generate J_1 arrives with $p_1 = 1$ and $g_1 = 2$. If job J_1 is assigned to the first machine, we generate J_2 with $p_2 = 1$, $r_2 = \varepsilon$(ε sufficiently small) and $g_2 = 1$, then the competitive ratio $C^A/C^* \approx 2/1 = 2$. If J_1 be assigned to machine M_2, we generate J_3 with $p_3 = 1+\lambda$, $r_3 = \varepsilon$ and $g_3 = 2$, the competitive ratio $C^A/C^* = (1+\lambda)/1 = 1+\lambda$ if job J_3 is assigned to machine M_2; if J_3 is assigned to machine M_1 at time $a(a<1)$, we generate J_4 with $p_4 = 1+\lambda-a$, $r_4 = a+\varepsilon$ and $g_4 = 1$, the competitive ratio $C^A/C^* \approx (2+\lambda)/(1+\lambda) = 1+\lambda$. Above all, we can conclude that the lower bound is at least $1+\lambda = (\sqrt{5}+1)/2$.

2.3 An Online Algorithm of the Problem

Here, we show an online algorithm which is proved to have the optimal performance. In order to describe the algorithm (or for the algorithm Alg_P2 and Alg_P3) clearly, we make some parameters in Table 1, which will be used later.

First we give the update algorithm UA_P1 for decision moment. Here C_i is the time machine M_i become idle. Then, the online algorithm Alg_P1 is proposed based on UA_P1.

Algorithm UA_P1

Step 1 Let t be the current decision time, C_i be the time machine M_i become idle after the decision at time t, $A_1(t)$, $A_2(t)$ be updated after the decision at time t($A_1(t)$ must be empty after updated).

Step 2 If $A_2(t) = \varnothing$ and at least one of the two machines is idle, the decision moment is the arriving time of the next job (if all jobs have arrived, stop the algorithm);

Table 1 Parameters used in the proposed algorithm

Parameters	Definitions	Values
t	The decision time	A job's arriving time or completion time
$A_1(t)$	The ordered set of jobs with GoS 1 which aren't processed on machine M_1 before time t	The sequence of $A_1(t)$ is according to the rule of ERT (earliest releasing time first)
$A_2(t)$	The ordered set of jobs with GoS 2 which aren't processed before time t	The sequence of $A_2(t)$ is according to the rule of LPT (longest processing time first)
$S(t)$	The maximum time that one of the two machines is idle before time t	The larger idle time before time t, 0 if the two machines are always busy
$\|A_2(t)\|$	The number of elements in $A_2(t)$	Nonnegative integers
J_l^t	The last job of $A_2(t)$	A specified job
$L_{2,t}$	The complete time of machine M_2 when it finishes all the assigned jobs before time t	A time when M_2 finishes its jobs which are assigned before time t
C_i^A	The maximum completion time of machine M_i using the proposed algorithm	A time when M_i finishes all jobs assigned to it

Step 3 Else if both $A_1(t)$ and $A_2(t)$ do not decrease after the decision at time t, the decision moment is the smaller one of the arriving time of the next job and C_2 (if all jobs have arrived, stop the algorithm).

Step 4 Else the decision moment is the smaller time of time $\min(C_1,\ C_2)$ and the arriving time of the next job.

Algorithm Alg_P1

Step 1 Set the arriving time of the first job is zero, $t := 0$, $A_1(t) := \varnothing$, $A_2(t) := \varnothing$, $J_l^t := \varnothing$, $S(t) := 0$.

Step 2 Update $A_1(t)$, $A_2(t)$, t and $S(t)$ in turn. Then update $A_1(t)$, $A_2(t)$ again when there are new jobs arriving.

Step 3 If the two machines are both busy at time t and $A_1(t) = \varnothing$, go to Step 2. Else if $A_1(t) \cup A_2(t) = \varnothing$ and there exist jobs arriving after t, go to Step 2; once all the jobs are scheduled, end the algorithm.

Step 4 Else if $A_1(t) \neq \varnothing$, let all jobs of $A_1(t)$ be assigned to machine M_1, go to Step 2.

Step 5 Else if $A_2(t) \neq \varnothing$ and machine M_2 is idle, let the first job of $A_2(t)$ be assigned to machine M_2, go to Step 2.

Step 6 Else if $A_1(t) = \varnothing$ and machine M_1 is idle:

6.1. If $|A_2(t)| \geqslant 2$, let the second job of $A_2(t)$ be assigned to machine M_1.

6.2. Else $|A_2(t)| = 1$, assign job J_l^t to machine M_1 if $p_l^t/(L_{2,t} - S(t)) \leqslant (1+\sqrt{5})/2$. Go to Step 2.

2.4 Competitive Ratio of the Proposed Algorithm

Firstly, when an instance is scheduled by the proposed algorithm and the two machines are both idle at a time interval $[a, b]$, then we only need to consider the jobs arriving after time b by the following lemma.

Lemma 1 *For the proposed algorithm, if a jobs arrives at time t and all the jobs arrive before time t have been completed before t, then this algorithm has the optimal solution if it has the best scheduling decision for jobs arriving after time t (including time t).*

Proof For any optimal schedule of any instance, as a job arrives at time t and all other jobs arriving before time t are completed before time t, it is obviously that the two machines are idle at time t and the jobs arriving after time t cannot be processed before time t by the optimal algorithm.

We supposed that the two machines cannot be idle at the same time interval in the following analysis. Then, we give a lemma which shows that the total remaining processing time of jobs with GoS 1 and the remaining processing time of each job with GoS 2 are not too big. According to Lemma 1, we just need to prove the case that the two machines cannot be idle at the same time until all jobs are finished (Fig. 1).

Lemma 2 *For any instance* I*, the sum of total remaining processing time of unprocessed jobs with GoS* 1 *and the remaining processing time of the processing job on machine M_1 at time C^* is no more than λC^* at time C^*; the remaining processing time of each job with GoS* 2 *at time C^* is no more than λC^* at time C^*. Here $\lambda = (\sqrt{5}-1)/2$.*

Proof Without loss of generality, let $C^* = 1$. As the optimal offline algorithm has makespan 1, we denote the processing job on machine M_1 and the unprocessed jobs with GoS 1 is $J_i (i = 1, 2, \cdots, s)$, the total remaining processing time of these jobs is $p_1' + \sum_{i=2}^{s} p_i$ (p_1' represents the remaining processing time of J_1). If $p_1' + \sum_{i=2}^{s} p_i > \lambda$, then there must exist a job J_k which is processed on machine M_1 with $g_k = 2$ and $p_k \geqslant p_1' + \sum_{i=2}^{s} p_i > \lambda$. If the starting time of J_k is t, there must exist time t_0 that the

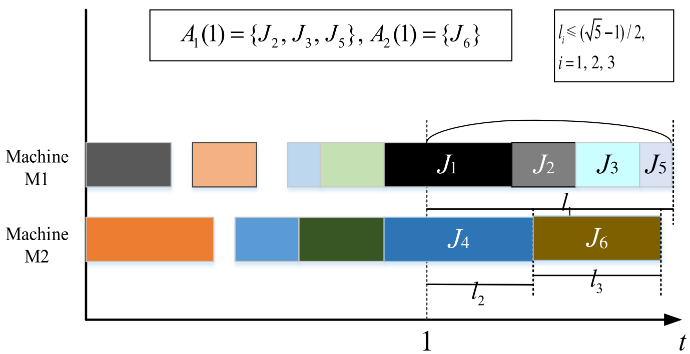

Fig. 1 Illustration for Lemma 2

two machines are both busy at time interval $[t_0,\ t_0 + 1 - \lambda]$ according to Step 6 of the algorithm. The total processing time of two machines is more than $1 - \lambda + 1 + \lambda = 2$, which contradicts the fact that makespan is 1.

For the remaining processing job $J_{1'}$ with GoS 2, if $J_{1'}$ is processing on machine M_1 in time 1, its remaining processing time is no more than λ according to the first part of proving. There are also two remaining cases:

Case 1′ $J_{1'}$ is processing on machine M_2 at time 1.

Let the remaining processing time of $J_{1'}$ be x(its total processing time is $p_{1'}$). Suppose that $x > \lambda$. Machine M_2 must be busy in time interval $[1 - p_{1'},\ 1]$ and machine M_1 must be busy in time interval $[1 - p_{1'},\ 1 - p_{1'} + x]$. Then we have $1 + x + x \leqslant 2$, e.g. $x \leqslant 1/2$, which is contracted with the assumption.

Case 2′ $J_{1'}$ is unprocessed at time 1.

With the same analysis with Case 1′, we can obtain $p_{1'} \leqslant \lambda$.
Using Lemma 2, we can get the competitive ratio of the proposed algorithm.

Theorem 2 *The competitive ratio of Algorithm Alg_P* 1 *is at most* $1 + \lambda$.

Proof Without loss of generality, let $C^* = 1$. There are two cases:

Case 1 $C_1^A \leqslant C_2^A$, e.g. $C^A = C_2^A$. If there just exists one job on machine M_2 which isn't completed at time 1, we have $C^A \leqslant 1 + \lambda$ according to Lemma 2. When there are at least two jobs processed on machine M_2, let the last job be J_k and $C_2^A > 1 + \lambda$. As $C^* = 1$, the arriving time of J_k is before time $1 - p_k$. Machine M_2 must be busy between time $1 - p_k$ and time $C_2^A - p_k$ according to Step 5 of Algorithm Alg_P1 and machine M_1 must be busy between time $1 - p_k$ and time $C_2^A - p_k$. Then, the total processing time of two machines is at least $C_2^A + (C_2^A - 1) > 2$, which contract with the fact that $C^* = 1$.

Case 2 $C_1^A > C_2^A$, e.g. $C^A = C_1^A$. If the last job J_k which is processed on machine M_1 has GoS 1, we have $C_1^A \leqslant 1 + \lambda$ according to Step 4 of Algorithm Alg_P1 and Lemma 2. If the GoS of J_k is 2 and $C_1^A > 1 + \lambda$, machine M_2 must be busy between time $1 - p_k$ and time $C_1^A - p_k$ as the arriving time of J_k is before time $1 - p_k$. Machine M_1 must be busy between time $1 - p_k$ and time $C_1^A - p_k$ according to Step 6. Then, the total processing time of two machines is at least $C_1^A + C_1^A - p_k > 2$, which contract with the fact that $C^* = 1$.

3 Optimal Algorithm for the Second Scheduling Problem

In this section, with the same contracture as Sect. 2, firstly we describe the second scheduling problem; then, the lower bound of this problem is analyzed; at last, we propose an online algorithm, which is proved to be optimal.

3.1 Problem Description

The considered problem can be described as: there are n different jobs J_i $(i = 1, \cdots, n)$, which have to be processed on one of the two parallel machines M_i $(i = 1, 2)$ without preemption. The GoS of machine M_i is i. Jobs are arrived over time with GoS (1 or 2). The two machines can be idle at the same time only when all the jobs arrived are completed. The jobs will be processed on a first-come-first-serviced (FCFS) basis. The notations are the same with Sect. 2.1. Then, the second problem can be described as $P2|\text{online},\ r_i,\ M(\text{GoS}),\ \text{no-delay1, FIFS}|C_{\max}$. To get the minimum makespan, we need to decide the assignment of each job after it arrives. As the second problem add a new requirement FCFS compared with the first one, with the same analysis as Theorem 1, we can obtain the lower bound of the second problem.

Theorem 3 *Any online algorithm for the second problem has a competitive ratio of at least* $(\sqrt{5}+1)/2$.

3.2 An Online Algorithm of the Second Problem

In this subsection, an online algorithm is proposed based on the algorithm of Sect. 2.3. The notations are the same as subsection 2.3 except $A_2(t)$, which is defined as: The set of jobs with GoS 2 which are not processed on the two machines before time t, the sequence of $A_2(t)$ is according to the rule of ERT (earliest releasing time first).

The proposed algorithm is almost the same as the algorithm in subsection 2.3 except Step 6.

Algorithm Alg_P2

Step 1 Set the arriving time of the first job is zero, $t := 0$, $A_1(t) := \varnothing$, $A_2(t) = \varnothing$, $J_l^t := \varnothing$ and $S(t) := 0$.

Step 2 Update $A_1(t)$, $A_2(t)$, t and $S(t)$ in turn. Then update $A_1(t)$, $A_2(t)$ again when there are new jobs arriving.

Step 3 If the two machines are both busy at time t and $A_1(t) = \varnothing$, go to Step 2. Else if $A_1(t) \cup A_2(t) = \varnothing$ and there exist jobs arriving after t, go to Step 2; once all the jobs are scheduled, end the algorithm.

Step 4 Else if $A_1(t) \neq \varnothing$, let all jobs of $A_1(t)$ be assigned to machine M_1, go to Step 2.

Step 5 Else if $A_2(t) \neq \varnothing$ and machine M_2 is idle, let the first job of $A_2(t)$ be assigned to machine M_2, go to Step 2.

Step 6 Else if $A_1(t) = \varnothing$ and machine M_1 is idle:

6.1. If $p_a^1/(L_{2,t} + \sum_{i\in I, i\neq 1} p_a^i - S(t)) \leqslant (\sqrt{5}+1)/2$, assign J_a^1 to machine M_1. Here p_a^i is the processing time of J_a^i which is the i-th element of set $A_2(t)$. Go to Step 2.

3.3 Competitive Ratio of the Proposed Algorithm

It is obviously that Lemma 1 is also correct for the second problem. The following analysis supposes that the two machines won't be both idle at the same time until all jobs are completed.

Lemma 3 *If J_n is processed on machine M_2, we have $C/C^* \leqslant (\sqrt{5}+1)/2$.*

Proof Without loss of generality, let $C^* = 1$. Note that $\lambda = (\sqrt{5}-1)/2$. As J_n is the last job on machine M_2, then $S_n = C - p_n$. Suppose that $C > 1+\lambda$, we have $r_n \leqslant 1 - p_n$ and machine M_2 must be busy between time interval $[1-p_n, C-p_n]$. If machine M_1 is busy between time interval $[1-p_n, 1+\lambda^2-p_n]$, then the total processing time of the two machines satisfies $\lambda^2 + C > 2$. We let the first idle time of machine M_1 after time $(1-p_n)$ is $1-p_n+x$ $(0 \leqslant x < \lambda^2)$, so $A_1(1-p_n+x) = \varnothing$. Let the processing job at time $1-p_n+x$ on machine M_2 be J_a and the first job of $A_2(1-p_n+x)$(after assigning job J_a)be J_k. If $S_k < C_a$, then J_k is processed on machine M_1 and $p_k > (C-1+p_n)/\lambda \geqslant 1$ according to Step 6 of the proposed algorithm. Then, we can obtain that $S_k = C_a$. If J_n and J_k are the same job, then $p_n > (C-1)/\lambda \geqslant 1$. If J_n and J_k are different jobs, then $p_k > (C-1+p_n-p_k)/\lambda$, i.e. $p_k > \lambda^2$. Then, we can obtain machine M_1 is busy between time interval $[S_k, C_k]$, so the total processing time of the two machines is more than 2. So we can conclude that $C \leqslant (1+\lambda)C^*$.

Lemma 4 *If J_n is processed on machine M_1, we have $C/C^* \leqslant (\sqrt{5}+1)/2$.*

Proof Without loss of generality, let $C^* = 1$. Note that $\lambda = (\sqrt{5}-1)/2$. Suppose that $C > 1+\lambda$. There are two cases:

Case 1 The GoS of J_n is 2.

We have $C - L \leqslant p_n$, so machine M_2 must be busy at time interval $[1-p_n, C-p_n]$. If machine M_1 is also busy at the above time interval, it can be easily obtained that the total processing time of the two machines is more than 2. We let the first idle time of machine M_1 after time $(1-p_n)$ be $1-p_n+x$. Let the processing job on machine M_2 at time $1-p_n+x$ be J_a and the first job of $A_2(1-p_n+x)$ (after assigning job J_a) be J_k. If J_n and J_k are the same job, we can obtain that $p_n > (C-1)/\lambda \geqslant 1$. If J_n and J_k are different jobs, then $p_k > (C-1+p_n-p_k)/\lambda$, i.e. $p_k > \lambda^2$. If J_k is processed on machine M_1, the total processing time of the two machines is more than 2, so J_k is processed on machine M_2. Then, we have machine M_1 is busy between time interval $[S_k, C_k]$, which can also result that the total processing time of the two machines is more than 2.

Case 2 The GoS of J_n is 1.

If there are no jobs with GoS 2 processed on machine M_1, we have $C = C^* = 1$. Let the last job with GoS 2 processed on machine M_1 be J_k, so $C - C^* \leqslant p_k$, i.e.$p_k > \lambda$. According to Step 6, we can obtain that the total processing time of the two machines is more than 2.

Theorem 4 *The competition ratio of the proposed algorithm for the first problem is* $(\sqrt{5}+1)/2$.

Proof Combing Lemmas 3 and 4.

4 Optimal Algorithm for the Third Problem

In this section, with the same contracture as Sects. 2 and 3, firstly we describe the third scheduling problem; then, the lower bound of this problem is analyzed; at last, we propose an online algorithm, which is proved to be optimal.

4.1 Problem Description

The considered online scheduling problem can be described as: there are n different jobs J_i $(i = 1, \cdots, n)$ which have to be processed on one of the two parallel machines M_i $(i = 1, 2)$ without preemption. The GoS of machine M_i is i. Jobs are arrived over time with GoS (1 or 2). A job must be assigned to one of the two machines as soon as it arrives, which can be denoted as $no - delay2$. Then, we can describe the second problem as $P2|online,\ r_i,\ M(GoS),\ no - delay2|C_{\max}$. With the same mark as Sect. 2.1, a job which is to be processed can be denoted by $(J,\ r,\ p,\ g)$. To get the minimum makespan, we need to decide the assignment of each job as soon as it arrives.

4.2 The Lower Bound of the Third Problem

In this subsection, we give the lower bound of any online algorithm for the second problem.

Theorem 5 *. Any online algorithm for the second problem has a competitive ratio of at least* $5/3$.

Proof C^* represents the optimal solution and C^A represents the solution of an online algorithm. At time $t = 0$, J_1 arrives with $p_1 = 1$ and $g_1 = 2$. If J_1 is scheduled on machine M_1, we generate J_2 with $p_2 = 1$, $g_2 = 1$ and $r_2 = \varepsilon$(sufficiently small), and its competitive ratio is $C^A/C^* \approx 2/1 = 2$. If J_1 is scheduled on machine M_2, we generate J_3 with $p_3 = 1$, $g_3 = 2$ and $r_3 = \varepsilon$. With the same analysis, J_3 should be scheduled to machine M_1(otherwise its competitive ratio is 2 when scheduled to machine M_2). We generate J_4 with $p_4 = 1$, $g_4 = 2$ and $r_4 = 2\varepsilon$. If J_4 is scheduled to machine M_1, we generate J_5 with $p_5 = 3$, $g_5 = 1$ and $r_5 = 3\varepsilon$, which has competitive ratio $C^A/C^* \approx 5/3$. If J_4 is scheduled to machine M_2, we generate J_6 with $p_6 = 3$, $g_6 = 2$ and $r_5 = 3\varepsilon$. If J_6 is scheduled to machine M_2, it has competitive ratio $C^A/C^* \approx 5/3$. If J_6 is scheduled to machine M_1, we generate J_7 with $p_7 = 6$, $g_7 = 1$ and $r_5 = 4\varepsilon$, which has competitive ratio $C^A/C^* \approx 10/6 = 5/3$.

4.3 An Online Algorithm for the Third Problem

Here, we show an online algorithm which is proved to have the optimal competitive ratio. The notations are almost the same with Sect. 2.3 except that $A_i(t)$ does not need to be sequenced. The decision moment is the arriving time of each job.

Algorithm Alg_P3

Step 1 Set the arriving time of the first job is zero, $t := 0$, $A_1(t) := \varnothing$, $A_2(t) = \varnothing$, $J_l^t := \varnothing$, $S(t) := 0$, $C_1^A := 0$, $C_2^A := 0$.
Step 2 Update t, $A_2(t)$, $A_1(t)$ and $S(t)$ in turn.
Step 3 If $A_1(t) \neq \varnothing$, put all jobs in $A_1(t)$ to machine M_1 and update C_1^A.
Step 4 While $A_2(t) \neq \varnothing$
Step 5 Pick the first job J_k in $A_2(t)$ and delete it from $A_2(t)$.
Step 6 If $C_1^A \geqslant C_2^A$ or machine M_2 is idle or $C_1^A - S(t) + p_t > 2(C_2^A - S(t))$, put J_k to machine M_2 and update C_2^A.
Step 7 Else, put J_k to machine M_1 and update C_1^A.
Step 8 end
Step 9 Go to Step 2 until all jobs are scheduled.

4.4 Competitive Ratio of the Proposed Algorithm

It is apparently that Lemma 1 is also correct for the third problem, so we suppose that the two machines won't be both idle at the same time interval. To simplify the analysis, Lemma 5 is used to limit the range of instances.

Lemma 5 *If the competitive ratio of Algorithm Alg_P*3 *is* α *when we consider all instances with the condition that there is at most one job arrived at any time, its competitive ratio must be* α *for all instance* (*without the condition*).

Proof For any instance I, we add $i\varepsilon_k$ to r_i, $i = 1, 2 \cdots, n$, and $\varepsilon_k \to 0$, $k \to \infty$. When $k \to \infty$, the new instance I_k satisfies the condition that there is at most one job arrived at any time. It is obviously that Lemma 5 is correct.

For any instance (the following analysis just considers the instances with the above condition), we cannot have the structure the jobs with GoS 1 are processed before the jobs with GoS 2 after time C^* on machine M_1. The following lemma just considers the jobs on machine M_2.

Lemma 6 *For any instance* I*, the remaining processing time of each job on machine* M_2 *at time* C^* *is no more than* $2C^*/3$ *at time* C^*.

Proof Without loss of generality, let $C^* = 1$. For any instance I, suppose that the remaining processing time $(p_k^{'})$ of J_k on machine M_2 at time C^* is more than $2/3$ at time 1. As $r_k \leqslant 1 - p_k$ and the starting time of J_k is $S_k > 5/3 - p_k$, then the two machines must be both busy at time interval $[1 - p_k, 4/3 - p_k]$ according to Steps 6 and 7. Then, we get the total processing time of all jobs must be more than 2, i.e. $C_2^A + 1/3 > 2$, which contract with the fact that $C^* = 1$.

Then, the competitive ratio of Algorithm Alg_P3 is proved to be 5/3 by Theorem 6.

Theorem 6 *The competitive ratio of Algorithm Alg_P3 is at most* $5/3$ *for the third problem.*

Proof Let $C^* = 1$. There are two cases:

Case 1 $C_1^A \leqslant C_2^A = C^A$.

We denote the last completed job as J_l. If there is only an uncompleted job on machine M_2 at time 1, we have $C^A \leqslant 5/3$ according to Lemma 6. When there are at least two uncompleted jobs on machine M_2 at time 1, suppose that $C_2^A > 5/3$. As $r_l \leqslant 1 - p_l$ and the starting time of J_l is $C_2^A - p_l$, we have $[1 - p_l, 4/3 - p_l]$ according to Steps 6 and 7 of Algorithm Alg_P3. Then, we get the total processing time of all jobs must be more than 2, i.e. $C_2^A + 1/3 > 2$, which contract with the fact that $C^* = 1$.

Case 2 $C_2^A < C_1^A = C^A$.

Let the last job processed on machine M_1 be J_l. If the GoS of J_l is 2 and $C_1^A > 5/3$, we can conclude that machine M_1 and M_2 are both busy at time interval $[1 - p_l, C_1^A - p_l]$. Then, we get the total processing time of all jobs must be more than 2, i.e. $sum \geqslant C_1^A + 2/3 > 2$, which contract with the fact that $C^* = 1$. If the GoS of J_l is 1 and $C_1^A > 5/3$, machine M_1 processes jobs with GoS 2 with total processing time more than 2/3 (put these jobs to set B). For the first processed job in set B, denoted as J_1, machine M_2 must process jobs with total processing time at least $p_1/2$. For the second processed job in set B, if machine M_2 is busy at time interval $[S_1, S_2]$(S_i is the starting time of J_i), then the total processing time of machine M_2 is at least $(p_1 + p_2)/2$ according to Steps 6 and 7; else, the total processing time is at least the sum of two parts, i.e. $(p_1 + p_2)/2$. Then, we obtain that the total processing time of machine M_2 is at least 1/3. As the processing time that we compute on machine M_2 makes sure that the two machines are both busy, the total processing time of all jobs must be more than 2, i.e. $sum \geqslant C_1^A + 1/3 > 2$, which contract with the fact that $C^* = 1$.

5 Conclusion and Future Research

In this paper, we investigate three online scheduling problems on two parallel identical machines under a grade of service provision. The jobs arrive over time in all the three problems. The first problem requires that the two machines cannot be idle at the same time until all arrived jobs have been processed, and the second problem further requires that jobs are processed by the rule of FCFS, while the third problem has the requirement that each job must be assigned to one of the two machines as soon as it arrives. We propose optimal algorithms for all the three problems. The three algorithms and the analysis are very novel and simple.

The future research mainly includes two parts: (1) the algorithm designing can be used for both two types of online scheduling problem: over time or over list; (2) the proving method can be used to analysis the competitive ratio of online scheduling problem.

Appendix A: Notations

In "Appendix A", some common notations in this article are introduced in detail, especially notations in the section of introduction.

$\alpha|\beta|\gamma$: the 3-field notation proposed by Graham et al. [2]. This notation can uniquely characterize a scheduling problem. α represents the machine environment, which contains only one item. β represents processing features and constraints, etc. γ describes minimization goal. In this paper, α includes the following situations:

- Pm — m parallel machines. Here m machines have the same speed. $m = 2$ means there are two available machines;
- Qm — m uniform machines. Here m machines have the different speed. $m = 2$ means there are two available machines;
- β — field may include multiple processing features and constraints. In this paper, we use the following notations;
- $online$ — represents that each job information is known only after it arrivals, which is the antonym of offline. Semi-online means that some information is known such as maximum processing time of all jobs. Semi-online scheduling need to add the additional information in β field such as using max when maximum processing time of all jobs is known in advance;
- M_j — machine eligibility constraints, i.e., not all machines can process job j. It means that each job must be processed in some machines. $M_j(GoS)$ is a special case of M_j, which means that a job can only be processed in the machine whose grade is not higher than the its grade;
- r_j — release time. It means that a job j cannot be processed before time r_j;
- $pmtn$ — preemption is allowed, which means that the jobs can be rescheduled to other machines even when they have not been finished;
- γ — is used to describe the minimizing objectives. In this paper, only makespan is involved;
- $C_{\max}$ — makespan, its definition is $\max(C_1, C_2, \cdots, C_n)$.

Appendix B: Competitive Ratios of Online Scheduling

In "Appendix B", we mainly introduce the research situation of the related problems in this paper (online over times and not batch scheduling).

Scheduling problem	Reference	Competitive ratio	Best ratio?
$P2\|\text{online}, r_j, M_j, p_j = p\|C_{\max}$	[15]	1.618 0	Yes
$P2\|\text{online}, r_j, M_j(\text{GoS}), p_j = p\|C_{\max}$	[15]	1.414 2	Yes
$P2\|\text{online}, r_j, M_j(\text{GoS})\|C_{\max}$	[16]	1.555 0	Yes
$Q2\|r_j, \text{online}\|C_{\max}$	[17]	1.618 0	No
$Pm\|r_j, \text{online}\|C_{\max}$	[17]	1.359 6	No
$P2\|\text{online}, r_j\|C_{\max}$	[13]	1.382	Yes
$P2\|\text{online}, r_j, pmtn\|C_{\max}$	[14]	1	Yes

References

[1] Mizgier, K.J., Wagner, S.M., Juettner, M.P.: Disentangling diversification in supply chain networks. Int. J. Prod. Econ. **162**, 115–124 (2015)

[2] Lee, K., Leung, J.Y.T., Pinedo, M.L.: Makespan minimization in online scheduling with machine eligibility. Ann. Oper. Res. **204**(1), 189–222 (2013)

[3] Graham, R.L., Lawler, E.L., Lenstra, J.K., et al.: Optimization and approximation in deterministic sequencing and scheduling: a survey. Ann. Discrete Math. **5**, 287–326 (1979)

[4] Park, J., Chang, S.Y., Lee, K.: Online and semi-online scheduling of two machines under a grade of service provision. Oper. Res. Lett. **34**(6), 692–696 (2006)

[5] 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**(3), 309–314 (2006)

[6] 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**(1), 367–371 (2012)

[7] Zhang, A., Jiang, Y., Tan, Z.: Online parallel machines scheduling with two hierarchies. Theor. Comput. Sci. **410**(38–40), 3597–3605 (2009)

[8] Tan, Z., Zhang, A.: A note on hierarchical scheduling on two uniform machines. J. Comb. Optim. **20**(1), 85–95 (2010)

[9] Jiang, Y.: Online scheduling on parallel machines with two GoS levels. J. Comb. Optim. **16**(1), 28–38 (2008)

[10] Liu, M., Xu, Y., Chu, C., et al.: Online scheduling on two uniform machines to minimize the makespan. Theor. Comput. Sci. **410**(21–23), 2099–2109 (2009)

[11] Mandelbaum, M., Shabtay, D.: Scheduling unit length jobs on parallel machines with lookahead information. J. Sched. **14**(4), 335–350 (2011)

[12] Lu, X., Liu, Z.: Semi-online scheduling problems on two uniform machines under a grade of service provision. Theor. Comput. Sci. **489**, 58–66 (2013)

[13] Chen, B., Vestjens, A.P.A.: Scheduling on identical machines: how good is LPT in an on-line setting? Oper. Res. Lett. **21**(4), 165–169 (1997)

[14] Noga, J., Seiden, S.S.: An optimal online algorithm for scheduling two machines with release times. Theor. Comput. Sci. **268**(1), 133–143 (2001)

[15] Hong, K.S., Leung, J.Y.T.: On-line scheduling of real-time tasks. In: Real-Time Systems Symposium. Proceedings, pp. 244–250. IEEE (1988)

[16] Lee, K., Leung, J.Y.T., Pinedo, M.L.: Scheduling jobs with equal processing times subject to machine eligibility constraints. J. Sched. **14**(1), 27–38 (2011)

[17] Xu, J., Liu, Z.H.: An optimal online algorithm for scheduling on two parallel machines with GoS eligibility constraints. J. Oper. Res. Soc. China **4**(3), 371–377 (2016)

[18] Li, S., Zhang, Y.: On-line scheduling on parallel machines to minimize the makespan. J. Syst. Sci. Complex. **29**(2), 472–477 (2016)

[19] Yuan, J.J., Ren, L.L., Tian, J., et al.: Online scheduling on two uniform unbounded parallel-batch machines to minimize makespan. J. Oper. Res. Soc. China **7**(2), 303–319 (2019)

[20] Cai, S., Liu, K.: Heuristics for online scheduling on identical parallel machines with two GoS levels. J. Syst. Sci. Complex. **32**(4), 1180–1193 (2019)