J Syst Sci Complex (2019) 32: 1180–1193

Heuristics for Online Scheduling on Identical Parallel Machines with Two GoS Levels*

CAI Shuang · LIU Ke

DOI: 10.1007/s11424-019-7427-6
Received: 15 December 2017 / Revised: 26 April 2018

Abstract This paper considers the online scheduling problem on m ($m \geq 3$) parallel machines (the first k machines with grade 1 and the remaining $m-k$ machines with grade 2) with two GoS levels and makespan as the objective function. The jobs arrive over time with grade 1 or 2 and an arrival job can be assigned to a machine only when the grade of the job is no less than the grade of the machine. Three cases are considered: (i) For $k = 1$, the authors present an online algorithm with competitive ratio of 9/5. (ii) For $1 < k < m - 1$, an online algorithm with competitive ratio of 2.280 is proposed. (iii) For $k = m - 1$, an online algorithm is presented with competitive ratio of 2. All the three algorithms are based on greedy algorithm with a similar structure. At last, numerical instances are given and the average competitive ratios of the instances show good performance of the proposed algorithms.

Keywords Grade of Service (GoS) constraints, heuristic algorithm, online scheduling, parallel machine scheduling.

1 Introduction

Scheduling problems are very common in many industries, such as industrial manufacturing, CPU operating. As one of the most important categories in scheduling, parallel machine scheduling has been studied for over fifty years. Many of the parallel machine scheduling problems are proved to be NP hard[1]. Therefore, polynomial algorithms to obtain the optimal solutions are not likely to be found, so many heuristics and meta-heuristics have been studied, such as Franca, et al.[2], Cheng and Gen[3], Chen and Powell[4], Chang, et al.[5]. Machine eligibility constraints are very common for the scheduling on parallel machines, which means that a

CAI Shuang (Corresponding author)
Logistics R&D Department, Beijing Jingdong Zhenshi Information Technology Co., Ltd. Beijing 100176*, China; Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing* 100190*, China; University of Chinese Academy of Sciences, Beijing* 100190*, China.* Email: caishuang@amss.ac.cn.
LIU Ke
Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100190*, China; University of Chinese Academy of Sciences, Beijing* 100190*, China.* kliu@amss.ac.cn.
*This research was supported by the National Natural Science Foundation of China under Grant Nos. 71390334 and 11271356.
◇ *This paper was recommended for publication by Editor WANG Shouyang.*

job cannot be processed on all parallel machines. For the parallel machine scheduling problem with releasing time and machine eligibility constraints, Grisselle and Armacost[6] proposed a heuristic algorithm to minimum the maximum lateness. As a special case of machine eligibility constraints, the Grade of Service (GoS) eligibility constraint is a qualitative concept, which is motivated by the scheme that the customer with higher lever should get better service. GoS eligibility constraint means that each job and machine can be graded and a job can be processed on a machine if the grade of the job is no less than the grade of the machine. Though many papers have been published for the parallel machine scheduling problems with different kinds of machine eligibility constraints, such as Grisselle and Armacost[6], Liao and Sheen[7], Edis and Ozkarahan[8], Tseng, et al.[9], few papers have been published for the parallel machine scheduling problems with GoS eligibility constraints under offline version.

While in many cases, the problems have the constraint that jobs are presented to schedulers only when they arrive, which are called online scheduling problems. There are two paradigms of online scheduling: Jobs arrive one by one or over time, i.e., online over list or online over time. Compared with the offline scheduling problems on parallel machines, there are really few papers studying the online scheduling problems on parallel machines. Competitive ratio is used to measure the performance of online algorithm, which is defined as the worst competitive ratio among all the instances, i.e., $c_A = \max_I C_I{}^*/C_I^A$, where c_A, I, C_I^* and C_I^A represent the competitive ratio of algorithm A, an instance, the makespan of an optimal offline schedule for instance I and the makespan by algorithm A for instance I, respectively[8]. For the paradigm of online over list, a job will arrive only when the former one is scheduled (i.e., jobs arrive one by one) and the arriving time of each job is 0. There are some papers studying for online over list in parallel machine scheduling, such as Jongho, et al.[10], Jiang, et al.[11], Yong, et al.[12], Zhang, et al.[13], Tan and Zhang[14], Jiang[15], Liu, et al.[16], Marvin and Shabtay[17].

For the paradigm of online over time, jobs arrive over time and a job can be seen and scheduled only when it arrives. Although the paradigm of online over time is more realistic, very little work has been published until now. For the problem $P2\,|online, r_j\,, pmtn\,|C_{\max}$ (preemption is allowed), Hong and Leung[18] provided an exact algorithm with competitive ratio of 1. Chen and Vestjens[19] studied problem $Pm\,|online, r_j\,|C_{\max}$ and proved that the algorithm of LPT has 3/2-competitive performance and the best competitive ratio is at least 1.347. John and Seiden[20] proposed an optimal online algorithm with competitive ratio of 1.382 when $m = 2$. Grissele and Armacost[21] gave an LPT-based heuristic algorithm for problem $Pm\,|online, r_j\,, M_j\,|C_{\max}$. Lee, et al.[22] proposed optimal algorithms for two problems $P2\,|online, r_j\,, M_j, p_j = p\,|C_{\max}$ and $P2\,|online, r_j\,, M_j(GoS), p_j = p\,|C_{\max}$ with competitive ratio of $(\sqrt{5}+1)/2$ and $\sqrt{2}$, respectively. For the online scheduling problem with arbitrary processing time of each job, i.e., $P2\,|online, r_j\,, M_j(GoS)\,|C_{\max}$, Xu and Liu[23] proposed an optimal algorithm with competitive ratio 1.555. The proposed algorithm allows that the two machines are both idle at the same time while not all arrival jobs have been processed. Li and Zhang[24] studied two online scheduling problems $U2\,|online, r_j\,|C_{\max}$ and $Pm\,|online, r_j\,|C_{\max}$. Recently, Cai, et al.[25] studied the online scheduling on two identical parallel machines under a grade of service provision. The paper gave a heuristic online

algorithm based on greedy algorithm, with competitive ratio 5/3.

There is a famous result proposed by Shmoys, et al.[26] which can be used as a general methodology for solving various online problems. The authors proved that any offline algorithm for a scheduling problem can be used to obtain an online algorithm for the problem where jobs arrive over time. As Ou, et al.[27] proposed the polynomial-time approximation scheme (PTAS) for offline problem $Pm\,|r_j\,, M_j(GoS)\,|C_{\max}$, the online scheduling problem $Pm\,|online, r_j\,, M_j(GoS)\,|C_{\max}$ has a $(2+\varepsilon)$-competitive polynomial time algorithm for any positive constant ε. However, the PTAS has very high running time, there are faster approximation algorithms with a constant worst-case bounds 4/3 for the offline problem $Pm|r_j, M_j(GoS)|$ $C_{\max}$ proposed by Ou, et al.[27] and Huo and Joseph[28]. Therefore, the effective online algorithm for the online problem $Pm\,|online, r_j\,, M_j(GoS)\,|C_{\max}$ is 8/3-competitive. And to our best knowledge, there are no other papers published for the online scheduling on more than two parallel machines with jobs arriving over time under GoS constraints.

Greedy algorithm means that the decision makers make the locally optimal choice at each stage with the hope of finding a global optimum. Cai, et al.[25] proposed an online scheduling algorithm based on greedy method for two parallel machines. In this paper the problem is harder and the online algorithms are more complex. We study the online scheduling problem on m $(m > 2)$ parallel machines with 2 GoS levels where jobs arrive over time. There are k machines with grade 1 and $m-k$ machines with grade 2. Each job has grade 1 or 2 and it can be assigned to a machines only when the grade of the job is no less than the grade of the machine. Three cases are considered: For $k = 1$, we proposed an online algorithm Alg1 with competitive ratio of 9/5. For $1 < k < m - 1$, an online algorithm Alg2 is presented to schedule jobs with competitive ratio of 2.280. For $k = m-1$, we propose an online algorithm Alg3 with competitive ratio of 2. The three algorithms have similar structures, which are very useful and novel for the considered problem. At last, a new measurement is presented: The average competitive ratio, which is defined as the average competitive ratio for all instances. Numerous instances are computed and the results demonstrate that the proposed algorithms have good performances on average. The mixed integer liner programming model for the problem $Pm\,|r_j\,, M_j(GoS)\,|C_{\max}$ is formulated and it is used to obtain the optimal solution for the offline scheduling version.

The contributions of this paper include the following two aspects: 1) The problem $Pm\,|r_j\,, M_j$ $(GoS)\,|C_{\max}$ is studied for the first time. 2) Three online algorithms are proposed for the three cases of the considered problem with effective competitive ratios and the average competitive ratios of them is very small through lots of experiments.

The rest of this paper is as follows. In Section 2, the considered problem is described with more details. In Section 3, online algorithm Alg1 is proposed with competitive ratio of 9/5 for the considered problem in the case of $k = 1$. Online algorithm Alg2 is presented for the case of $1 < k < m - 1$ in Section 4 and its competitive ratio is 2.280. In Section 5, online algorithm Alg3 is proposed with competitive ratio of 2 for the case of $k = m - 1$. In Section 6, for each of numerous instances, the offline optimal makespan and the makespan by the proposed algorithms are computed to obtain the competitive ratio. The results show the effectiveness of the three proposed online algorithms. Finally, we give the conclusions in Section 7.

2 Problem Description

In this paper, we consider the online scheduling on m ($m > 2$) identical parallel machines with two GoS levels. Suppose that the total number of parallel machines is $k+l$ (i.e., $m = k+l$) and the first k machines have grade 1, while the last l machines have grade 2, which can be denoted as 1-machines and 2-machines, respectively. Jobs arrive over time with grade 1 or 2, i.e., 1-jobs or 2-jobs. We do not know anything about each job until it arrives. For any instance I, the arriving (processing) time and the grade of job J_j are denoted as r_j (p_j) and g_j. The manager needs to decide the assignment of each job after it arrives in order to minimize the makespan. The considered problem with three cases can be described as the $\alpha\,|\beta\,|\gamma$ notation:

$$P_{1+m-1}\,|online, r_j, M_j\ (GoS)\,|C_{\max}\,,$$
$$P_{k+m-k}\,|online, r_j, M_j\ (GoS)\,|C_{\max}\,,$$
$$P_{m-1+1}\,|online, r_j, M_j\ (GoS)\,|C_{\max}\,.$$

Here the three cases have different machine environments:

1) P_{1+m-1} represents one 1-machine and $m-1$ ($m-1>1$) 2-machines.

2) P_{k+m-k} represents k ($k>1$) 1-machines and $m-k$ ($m-k>1$) 2-machines.

3) P_{m-1+1} represents $m-1$ ($m-1>1$) 1-machines and one 2-machine.

The constraints are the same for the three cases of the considered problem:

a) online: The information of any job cannot be known until it arrives.

b) r_j: The jobs arrive over time, i.e., each job has a release time.

c) $M_j(GoS)$: $M_j(GoS)$ represents the GoS eligibility constraints (M_j represents the machine eligibility constraints).

3 Problem $P_{1+m-1}\,|online, r_j, M_j\ (GoS)\,|C_{\max}$

In this section, the problem $P_{1+m-1}\,|online, r_j, M_j\ (GoS)\,|C_{\max}$ is considered. Firstly a new online algorithm Alg1 is proposed. Then, the time complexity of algorithm Alg1 is analyzed. At last, the competitive ratio of algorithm Alg1 is proved to be at most 9/5.

3.1 An Online Algorithm for Problem $P_{1+m-1}\,|online, r_j, M_j\ (GoS)\,|C_{\max}$

Here we show a new online algorithm which is proved to have a good performance. In order to describe the algorithm clearly, we make some definitions:

$\boldsymbol{t}$: The decision time, which can be the arriving time or completion time of each job.

$\boldsymbol{A_1}(t)$: The set of arrival 1-jobs which have not been processed on the 1-machines before time t, the sequence of $A_1(t)$ is sorted according to the rule of ERT (earliest releasing time first).

$\boldsymbol{A_2}(t)$: The set of arrival 2-jobs which have not been processed before time t, the sequence of $A_2(t)$ is sorted according to the rule of LPT (longest processing time first).

$|A_2(t)|$: The number of elements in $A_2(t)$.

$\boldsymbol{J_j^t}$: The j-th job of $A_2(t)$.

$\boldsymbol{C_{i,t}}$: The completion time of machine M_i when finishing all the assigned jobs before or at time t.

$\boldsymbol{C}^A$: The maximum completion time (makespan) among all jobs by the proposed algorithm.

$C_{i,t}$ can be understood as the time when machine M_i become idle (if no jobs are assigned to the machine after time t). The online algorithm Alg1 is proposed for each decision moment t (by pseudo code).

Algorithm Alg1

---

1) Update $A_1(t)$, $A_2(t)$ when there are new jobs arriving.
2) If $A_1(t) \neq \varnothing$, let all jobs of $A_1(t)$ be assigned to machine M_1.
3) While $A_2(t) \neq \varnothing$ and one of the 2-machines is idle
4) Let the first job of $A_2(t)$ be assigned to this 2-machine.
5) If machine M_1 is idle:
6) For $j = 1$ *to* $|A_2(t)|$:
7) If we assign all the jobs before J_j^t in $A_2(t)$ to 2-machines by greedy algorithm and
8) In this case $p_j^t / \min_{i>1}(C_{i,t} - S_{i,a}) \leq 5/4$ ($S_{i,a}$ is the starting time of job $J_{i,a}$ which
9) is processed on machine M_i at time t), assign job J_j^t to machine M_1, break.

Algorithm Alg1 is proposed as shown above, while it may be a bit complicated in lines 6–9. The greedy algorithm means that the job with the largest processing time will be assigned to the machine with the smallest completion time. Lines 6–9 is used to choose a 2-job J_j^t which has the largest processing time in $A_2(t)$ and satisfies $p_j^t / \min_{i>1}(C_{i,t} - S_{i,a}) \leq 5/4$. Here $C_{i,t}$ may be larger than the completion time of $J_{i,a}$ in machine M_i according to line 7.

The time complexity of Alg1 is analyzed as follows: The time for sorting the jobs is at most $O(n^2)$. the decision times are the arriving times or completion times of all jobs, i.e., $O(n)$, while the time for each decision is at most $O(n^2 m)$, so the total time complexity for decision is at most $O(n^3 m)$. We have the time complexity of Alg1 is $O(n^3 m)$.

3.2 The Competitive Ratio of Algorithm Alg1

In this subsection, we give the competitive ratio of the proposed algorithm. Firstly, we give the analysis for the case that the last completed job J_n is a 1-job in Lemma 3.1. Then, Lemma 3.2 deals with the case that the last completed job J_n is a 2-job. We denote the makespan by algorithm Alg1 as C^A and the optimal makespan under offline condition as C^*. We just need to consider the case that all machines are not idle at the same time until all jobs are completed by algorithm Alg1, because all jobs arriving before this time is completed at this time by Alg1 and all jobs arriving after this time cannot start before it in the optimal schedule. In the following Lemmas 3.1 and 3.2, the continuous assigned 1-jobs $J_{q_1}, J_{q_2}, \cdots, J_{q_s}$ by algorithm Alg1 are seen as a new 1-job J_q with $r_q = \min_{1 \leq i \leq s} r_{q_i}$ and $p_q = \sum_{1 \leq i \leq s} p_{q_i}$.

Lemma 3.1 *If the last completed job J_n is a 1-job, the competitive ratio of algorithm Alg1 is at most 9/5.*

Proof For any instance I, without loss of generality, let $C^* = 1$. Suppose that $C^A > 9/5$. According to algorithm Alg1, if there are no 2-jobs assigned to machine 1, the makespan by

Alg1 must be equal with the optimal makespan. Define the last 2-job on machine M_1 as J_w and the starting time of J_w is $S_w = C^A - p_n - p_w < 1 - p_n$. We have $p_w > 4/5$ and $r_w < 1/5$. In lines 6–9 of Alg1, a 2-job that is assigned to machine M_1 at time t must satisfy $p_w/\min_{i>1}(C_{i,t} - S_a) \leq 5/4$. Then there is at least a job with processing time more than 16/25 on each 2-machine, so the instance has at least $(l+1)$ 2-jobs with processing time of each job more than 16/25 and the processing time of the other jobs should be less than 9/25 because $C^* = 1$.

As job J_w should be assigned to machine M_1 before time $C^A - p_n - p_w$ if machine M_1 is idle at any time in time interval $(1-p_w, C^A - p_n - p_w)$, machine M_1 must process a 2-job or a 1-job at time $1 - p_w$. If machine M_1 only processes 1-jobs in time interval $(1 - p_w, C^A - p_n - p_w)$, the total processing time of all 1-jobs is at least $C^A - 1 > 9/25$, which is also contracted with $C^* = 1$. In the schedule of algorithm Alg1, if a 2-job J_r ($J_r \neq J_w$) starts after or at time $1-p_w$ on machine M_1, the processing time of J_r should be more than 16/25, which is contracted with $C^* = 1$, so no other 2-job starts after time $1 - p_w$. If a 2-jobs J_r starts before time $1 - p_w$ and is completed after $1 - p_w$ and we denote the sum of processing times of these 1-jobs which are processed in $(1-p_w, C^A - p_n - p_w)$ as x ($x \geq 0$), it satisfies that $p_r + x > C^A - 1 - p_n > 4/5 - p_n$ and $p_n + x < 9/25$, so we have $p_r > 11/25$, which is also contracted with $C^* = 1$. Above all, we have $C^A \leq 9/5$. ∎

In Lemma 3.2 and Lemma 4.1, the total processing time after time S by the optimal schedule includes at least two cases for each job J_j: 1) p_j if $r_j \geq S$. 2) $\max(0, r_j + p_j - S)$ if $r_j < S$.

Lemma 3.2 *If the last completed job J_n is a 2-job, the competitive ratio of algorithm Alg*1 *is at most* 9/5.

Proof For any instance I, without loss of generality, let $C^* = 1$. Suppose that $C^A > 9/5$. As J_n is a 2-job, then the 2-machines must be busy in time interval $[1-p_n, C^A - p_n]$. According to lines 6–9, if machine M_1 is idle at a time t in the above time interval, J_n (or a job before J_n in $A_2(t)$) should be assigned to M_1, so machine M_1 must be busy in time interval $[1 - p_n, C^A - p_n]$. We denote the last idle time of the 2-machines before time $1 - p_n$ as S, if $S \leq 4/5 - p_n$, it is obviously that the total processing time of all machines is more than $l + 1$, so we have $1 - p_n \geq S > 4/5 - p_n$.

If $p_n > 3/5$, there must exist a 2-job ($\neq J_n$) with processing time more than 3/5 on each 2-machine and some 1-jobs with the sum of the processing times at least 2/5 or a 2-job ($\neq J_n$) with processing time at least 2/5 on machine M_1, which is contracted with $C^* = 1$. Therefore, we have $p_n \leq 3/5$.

The continuous 1-jobs on machine M_1 can be seen as a 1-job in the following analysis of this lemma. The jobs which are processed on 2-machines and arrive before time S and are completed after time S are denoted as $J_{i_1}, J_{i_2}, \cdots, J_{i_u}$ and $u < l$. Let the maximum waiting time among the above jobs as x, i.e., $x = \max_j(S_{i_j} - r_{i_j})$. There are three cases:

i) If machine M_1 is not always busy in time interval $[S, 1 - p_n]$, the job processed after or at time $1 - p_n$ must be arrived after time S. As the schedule by Alg1 process $J_{i_1}, J_{i_2}, \cdots, J_{i_u}$ at or before time S, if $x \leq \frac{S}{2}$, the total processing time after time S by the optimal schedule is at

least $(C^A-p_n-S)*l+C^A-1+p_n-x*l \geq (1-S)*(l+1)+(C^A-1-p_n-\frac{S}{2})*l+C^A-2+\frac{S}{2}+p_n$. However, $(C^A-1-p_n-\frac{S}{2})*l+C^A-2+\frac{S}{2}+p_n \geq (C^A-\frac{3}{2}-\frac{p_n}{2})*l+C^A-2+\frac{1}{2}>0$, which is contracted with $C^*=1$. If $x>\frac{S}{2}$, the total processing time of all machines by Alg1 is at least $(C^A-p_n-S)*l+C^A-1+\frac{S}{2}*l+\frac{S}{2}+p_n \geq (C^A-p_n-\frac{S}{2})*l+C^A-1+\frac{S}{2}+p_n \geq (C^A-\frac{p_n}{2}-\frac{1}{2})*l+C^A-1+\frac{1}{2}>l+1$, which is contracted with $C^*=1$.

ii) If machine M_1 is always busy in time interval $[S,1-p_n]$ and processes a 1-job at time S, there are two cases with almost the same analysis of i): if $x \leq \frac{S}{2}$, the total processing time after time S by the optimal schedule is at least $(C^A-p_n-S)*(l+1)-\frac{S}{2}*l-S+p_n \geq (1-S)*(l+1)+(C^A-1-p_n-\frac{S}{2})*(l+1)-\frac{S}{2}+p_n$, by the same method of i) we have it is contracted with $C^*=1$. If $x>\frac{S}{2}$, we also have the total processing time of all machines by Alg1 is above $l+1$, which is contracted with $C^*=1$.

iii) If machine M_1 is always busy in time interval $[S,1-p_n]$ and processes a 2-job J_q at time S, there are two cases: a) Machine M_1 does not process a 1-job immediately after the completion time of J_q, by the same analysis of ii), we can obtain a contradiction. b) Otherwise, a 1-job J_w is processed once J_q is completed. If $r_w \geq S$ or $C_w \leq 1-p_n$, we can easily obtain a contradiction by the method of i) and ii), so $r_w<S$ and $C_w>1-p_n$. Although there are two jobs which arrive before time S and are completed after time S, we have in the optimal schedule, the starting time of J_q (J_w) is at least 0 (S_q). Therefore, by the same analysis of ii) we can obtain a contradiction with $C^*=1$. ∎

Theorem 3.1 *Algorithm Alg*1 *has competitive ratio of* 9/5 *for problem* $P_{1+m-1}\,|online, r_j, M_j\ (GoS)\,|C_{\max}$.

Proof Combining Lemma 3.1 and Lemma 3.2, we can obtain Theorem 3.1. It is clearly that the lower bound is tight. ∎

4 Problem $P_{k+m-k}\,|online, r_j, M_j\ (GoS)\,|C_{\max}$

In this section, the problem $P_{k+m-k}\,|online, r_j, M_j\ (GoS)\,|C_{\max}$ is considered. Firstly a new online algorithm Alg2 is proposed. Then, the time complexity of algorithm Alg2 is analyzed. At last, the competitive ratio of algorithm Alg2 is proved to be at most 2.280.

4.1 An Online Algorithm for Problem $P_{k+m-k}\,|online, r_j, M_j\ (GoS)\,|C_{\max}$

Here we show a new online algorithm for problem $P_{k+m-k}\,|online, r_j, M_j\ (GoS)\,|C_{\max}$ which is proved to have a good performance. We use the same definitions as Subsection 3.1 except the following two notations:

$\boldsymbol{A_1}(t)$: The set of arrival 1-jobs which have not been processed on the 1-machines before time t, the elements in $A_1(t)$ is sorted according to the rule of LPT (longest processing time first).

$\boldsymbol{S}(t)$: The maximum time that one of the 2-machines is idle before time t.

The online algorithm Alg2 for each decision moment t is proposed as follows (by pseudo code):

Algorithm Alg2

1) Update $A_1(t)$, $A_2(t)$ when there are new jobs arriving.
2) While $A_1(t) \neq \varnothing$ and one of the 1-machines is idle.
3) Let the first job of $A_1(t)$ be assigned to this 1-machine.
4) While $A_2(t) \neq \varnothing$ and one of the 2-machines is idle
5) Let the first job of $A_2(t)$ be assigned to this 2-machine.
6) Flag=1.
7) While one of the 1-machine is idle and Flag=1
8) Flag=0.
9) For $j = 1$ *to* $|A_2(t)|$:
10) If we assign all the jobs before J_j^t in $A_2(t)$ to 2-machines by greedy algorithm and
11) In this case $p_j^t/\min_{i>k}(C_{i,t} - S(t)) \leq (\sqrt{17}-1)/4$, assign job J_j^t to a idle 1-machine
12) and let Flag=1, break.

The lines 6–12 use the greedy algorithm similar to lines 5–9 of Alg1. The time complexity of Alg2 are analyzed as follows: The time for sorting the jobs is at most $O(n^2m)$. the decision times are the arriving times or completion times of all jobs ($O(n)$ times), while the time for each decision is at most $O(n^2m)$, so the total time complexity for decision is at most $O(n^3m)$, i.e., the time complexity of Alg2 is $O(n^3m)$.

4.2 The Competitive Ratio of Algorithm Alg2

In this subsection, we give the competitive ratio of the proposed algorithm Alg2. With the same analysis of Subsection 3.2, firstly we give the proof when the last completed job J_n is a 1-job in Lemma 4.1. Then, Lemma 4.2 deals with the case that the last completed job J_n is a 2-job.

Lemma 4.1 *If the last completed job J_n is a 1-job, the competitive ratio of algorithm Alg2 is at most $(\sqrt{17}+5)/4 \approx 2.280$.*

Proof For any instance I, without loss of generality, let $C^* = 1$ and $\alpha = (\sqrt{17}-3)/4$ and we denote the 1-machine which processes J_n as machine M_i. Suppose that $C^A > 2 + \alpha$. As the last job J_n is a 1-job, all 1-machines do not process 2-jobs after time $1 - p_n$ unless their starting times are before time $1 - p_n$. All 1-jobs have arrived at time $1 - p_n$ according to lines 2–3 and the definition of $A_1(t)$ and all 1-machines are busy at time interval $[1 - p_n, C^A - p_n]$. We denote the last completed 2-job on the 1-machines as J_{ls} (processed on machine $M_{i'}$) and it can be known that $S_{ls} < 1 - p_n$. As the optimal makespan for this instance is 1, we have $p_{ls} \leq 1$ and $C_{ls} = S_{ls} + p_{ls} < 2 - p_n < C^A - p_n$. All 1-machines must process 1-jobs at time interval $[C_{ls}, C^A - p_n]$ according to lines 2–3, so all 1-machines processed at least one 1-job with processing time more than p_n during time interval $[C_{ls} - p_n, C^A - p_n]$. We obtain k' 1-jobs ($k' \geq k$) with processing time above or equal to p_n at this time interval, so there are at least $(k'+1)$ 1-jobs with starting time after $1 - p_n$. While these $(k'+1)$ 1-jobs are started after

job J_{ls} and the optimal makespan is 1, the earliest arriving time among these jobs is at most $1-2p_n$ and $S_{ls} < 1-2p_n$, which can obtain that $p_n \leq 0.5$.

If there exists y ($y \geq 0$) 1-jobs with arriving time before S_{ls} and completion time after C_{ls}, the starting time of these jobs by algorithm Alg2 must be at or before time S_{ls}. The earliest arriving time among the 1-jobs with starting time after or at time S_{ls} is at most $1-(C^A-p_n-C_{ls})$, i.e., $S_{ls} \leq 1-(C^A-p_n-C_{ls})$, because the total processing time after S_{ls} by the optimal schedule is at least $(C^A-p_n-C_l)*k-(S_{ls}-p_{ls})*y \geq C^A-p_n-C_l)*k$ ($S_{ls} \leq p_{ls}$ as $(C^A-p_n-C_l)*k-(S_{ls}-p_{ls})*y \leq (1-S_{ls})*k$). We have $p_{ls} \geq C_{ls}-(1-(C^A-p_n-C_{ls})) = C^A-1-p_n > 1+\alpha-p_n \geq 0.5+\alpha$.

According to lines 6–12, J_{ls} is assigned to machine $M_{i'}$ at time S_{ls} if we assign all the jobs before J_{ls} in $A_2(S_{ls})$ to 2-machines by greedy algorithm and $p_{ls}/\min_{i>k}(C_{i,S_{ls}}-S(S_{ls})) \leq (\sqrt{17}-1)/4$. It is obvious that all the jobs before J_{ls} in $A_2(S_{ls})$ to 2-machines must be assigned to 2-machines as J_{ls} is the 2-job with the last completion time on all 1-machines. As $C_{i,S_{ls}}-S(S_{ls}) \geq p_{ls}/(\sqrt{17}-3)/4 \geq 1, \forall i > k$, the total processing time of all 2-machines must be at least l. While all the 1-machines are busy during time interval $[1-p_n, C^A-p_n]$, the total processing time of all machines is at least $k(1+\alpha)+l > m$, which is contracted with $C^* = 1$. ∎

Lemma 4.2 *If the last completed job J_n is a 2-job, the competitive ratio of algorithm Alg2 is at most $(\sqrt{17}+5)/4 \approx 2.280$.*

Proof For any instance I, without loss of generality, let $C^* = 1$ and $\alpha = (\sqrt{17}-3)/4$. Suppose that $C^A > 2+\alpha$. As the last job J_n is a 2-job, then all 2-machines must be busy at time interval $[1-p_n, C^A-p_n]$. According to lines 6–12, all the 2-jobs on 2-machines with starting time after $1-p_n$ have already arrived at time $1-p_n$ and they have the front position in $A_2(1-p_n)$ compared with job J_n. If a 1-machine is idle at time $[t_0, t_1], 1-p_n \leq t_0 < t_1 \leq C^A-p_n$, job J_n should be assigned to the idle 1-machine, so all 1-machines have to be busy at time interval $[1-p_n, C^A-p_n]$. In this way, we have the total processing time by algorithm Alg2 is above m, which is contracted with $C^* = 1$. ∎

Theorem 4.1 *Algorithm Alg2 for problem $P_{k+m-k}\,|online, r_j, M_j\ (GoS)\,|C_{\max}$ has competitive ratio of $(\sqrt{17}+5)/4 \approx 2.280$.*

Proof Combining Lemma 4.1 and Lemma 4.2, we can obtain Theorem 4.1. In fact, the competitive ratio of 2.280 is tight, which can be obtained as follows (Suppose $k = 4$ and $l = 2$):

i) At time $t = 0$ two 2-jobs J_1, J_2 arrives with $p_1 = p_2 = \frac{2}{\sqrt{17}+3}$. According to Alg2, J_1, J_2 are assigned to the two 2-machines.

ii) At time $t = \varepsilon$ (a sufficient small and positive number) three 2-jobs J_3, J_4, J_5 arrive with $p_3 = p_4 = p_5 = 1+\varepsilon$. According to lines 6–12 of Alg2, J_3, J_4, J_5 cannot be assigned to 1-machines if no new jobs arrive.

It is clearly that the competitive ratio of the above instance is infinitely close to 2.280 when $\varepsilon \to 0$. ∎

5 Problem $P_{m-1+1}\,|online, r_j, M_j\ (GoS)\,|C_{\max}$

In this section, the problem $P_{m-1+1}\,|online, r_j, M_j\ (GoS)\,|C_{\max}$ is considered. Firstly a new online algorithm Alg3 is proposed. Then, the time complexity of algorithm Alg3 is analyzed. At last, the competitive ratio of algorithm Alg3 is proved to be at most 2.

5.1 An Online Algorithm for Problem $P_{m-1+1}\,|online, r_j, M_j\ (GoS)\,|C_{\max}$

Here we show a new online algorithm for the third case of the scheduling problem which is proved to have a good performance. In Subsection 5.1, we use the same definitions of the notations as Subsection 4.1. The online algorithm Alg3 for each decision moment is proposed by pseudo code as follows.

Algorithm Alg3

---

1) Update $A_1(t)$, $A_2(t)$ when there are new jobs arriving.
2) While $A_1(t) \neq \varnothing$ and one of the 1-machines is idle
3) Let the first job of $A_1(t)$ be assigned to this 1-machine.
4) If $A_2(t) \neq \varnothing$ and the 2-machine is idle
5) Let the first job of $A_2(t)$ be assigned to the 2-machine.
6) Flag=1.
7) While one of the 1-machine is idle and Flag=1.
8) Flag=0.
9) For $j = 1$ *to* $|A_2(t)|$:
10) If we assign all the jobs before J_j^t in $A_2(t)$ to the 2-machine.
11) And in this case $p_j^t / (C_{m,t} - S(t)) \leq 1$, assign job J_j^t to a idle 1-machine.
12) And let Flag=1, break.

The time complexity of Alg3 are analyzed as follows: i) The time for sorting the jobs is at most $O(n^2)$. ii) The decision times are jobs' arriving times or completion times ($O(n)$), while the time for each decision is at most $O(n^2m)$. Therefore, the total time complexity for decision is at most $O(n^3m)$.

5.2 The Competitive Ratio of Algorithm Alg3

This subsection gives the competitive ratio of the proposed algorithm Alg3. With the same analysis of Sections 3 and 4, firstly the proof is given when the last completed job J_n is a 1-job. Then, Lemma 5.2 deals with the case that the last completed job J_n is a 2-job.

Lemma 5.1 *If the last completed job J_n is a 1-job, the competitive ratio of algorithm Alg3 is at most 2.*

Proof For any instance I, without loss of generality, let $C^* = 1$. Suppose that $C^A > 2$. We denote the 1-machine which processes J_n as machine M_i and the last completed 2-job on the 1-machines as J_{ls}. With the same analysis as Lemma 4.1, $p_{ls} \geq C_{ls} - (1 - (C^A - p_n - C_{ls})) = C^A - 1 - p_n > 1 - p_n$. According to lines 6–12, J_{ls} is assigned to a 1-machine at time S_{ls} when it satisfies: If we assign all the jobs before J_{ls} in $A_2(S_{ls})$ to the 2-machine and

$p_{ls}/(C_{m,S_{ls}} - S(S_{ls})) \leq 1$. It is obvious that all the jobs before J_{ls} in $A_2(S_{ls})$ to the 2-machine must be assigned to the 2-machine as J_{ls} is the 2-job with the last completion time on the 1-machines. While all the 1-machines are busy during time interval $[1 - p_n, C^A - p_n]$, the total processing time of all machines is at least $m - p_n + p_n > m$, which is contracted with $C^* = 1$. ∎

Lemma 5.2 *If the last completed job J_n is a 2-job, the competitive ratio of algorithm Alg3 is at most 2.*

Proof For any instance I, without loss of generality, let $C^* = 1$. Suppose that $C^A > 2$. As the last job J_n is a 2-job, then the 2-machine must be busy at time interval $[1 - p_n, C^A - p_n]$. According to Alg3, all the 2-jobs on the 2-machine with starting time after $1 - p_n$ have already arrived at time $1 - p_n$ and they have the more front position in $A_2(1 - p_n)$ compared with job J_n. If a 1-machine is idle at time $[t_0, t_1], 1 - p_n \leq t_0 < t_1 \leq C^A - p_n$, job J_n should be assigned to the idle 1-machine, so all 1-machines have to be busy at time interval $[1 - p_n, C^A - p_n]$. While in this way, it is obvious that the total processing time by algorithm Alg3 is above m, which is contracted with $C^* = 1$. ∎

Theorem 5.1 *Algorithm Alg3 for problem $P_{m-1+1}\,|online, r_j, M_j\ (GoS)\,|C_{\max}$ has competitive ratio of 2.*

Proof Combining Lemma 5.1 and Lemma 5.2, we can obtain Theorem 5.1. In fact, it is clearly that the lower bound 2 of Alg3 is tight. ∎

6 Experiments for Average Competitive Ratio

In this section, we build 100 instances for the scheduling problem and compute the competitive ratio for each instance in each of three machine environments.

6.1 Preparation for Experiments

Firstly, we give an introduction for the instances generating. All the instances are the same except the machine environment, which are $(1, 5)$, $(3, 5)$, and $(5, 1)$ (the first number represents the number of 1-machines, while the second one represents the number of 2-machines), respectively. The number of jobs is 20 for all the instances. Each instance is obtained by the following method: Both the processing time and the arriving time are subject to uniform distribution ($[1, 5]$ and $[0, 30]$, respectively). The grade of each job is 1 or 2 randomly with equal probability 1/2. In order to obtain the optimal schedule for problem $Pm\,|r_j\ , M(GoS)\,|C_{\max}$, we propose a mixed integer liner programming and the optimal makespan is obtained by the lpsolve solver (the optimal makespan of each instance can be obtained within 3 minutes). In the model, we

sort the jobs by arriving time, i.e., $r_1 \le r_2 \le \cdots \le r_n$.

$$\min \quad C_{\max} \tag{1}$$

$$\text{s.t.} \quad C_{\max} \ge C_{i,n}, \quad \forall i, \tag{2}$$

$$C_{i,j} \ge (r_j + p_j)X_{j,i}, \quad \forall i, j, \tag{3}$$

$$C_{i,j} \ge C_{i,j-1} + p_j X_{j,i}, \quad \forall i, j \ge 2, \tag{4}$$

$$X_{j,i} \le g_j - 1, \quad \forall j, \ i > k, \tag{5}$$

$$\sum_{1 \le i \le m} X_{j,i} = 1, \quad \forall j, \tag{6}$$

$$C_{\max} \text{ and } C_{i,j} : \text{Continuous variable}, X_{j,i} \in \{0, 1\}. \tag{7}$$

6.2 Analysis for the Experiments' Results

For each online algorithm A, the running results for algorithm A are compared with the optimal makespan for each instance and the ratios for all the 100 instances are summarized (the ratio of each instance = the makespan by A/the optimal makespan). Table 1 records the average ratios in group of ten instances. The results demonstrate the average ratio for the instances is very small and very close to 1. Although the instances are just some cases for the problems and they cannot represent the average ratio for all the cases, we can easily get the conclusion that the proposed algorithms (Alg1, Alg2 and Alg3) are very effective online algorithms for the scheduling problem.

Table 1 The results for the scheduling problem under three machine environments

Alg1		Alg2		Alg3	
Instances	Ratios	Instances	Ratios	Instances	Ratios
Problems 01–10	1.026	Problems 01–10	1.077	Problems 01–10	1.088
Problems 11–20	1.018	Problems 11–20	1.050	Problems 11–20	1.082
Problems 21–30	1.018	Problems 21–30	1.083	Problems 21–30	1.087
Problems 31–40	1.025	Problems 31–40	1.060	Problems 31–40	1.072
Problems 41–50	1.018	Problems 41–50	1.054	Problems 41–50	1.064
Problems 51–60	1.021	Problems 51–60	1.056	Problems 51–60	1.080
Problems 61–70	1.022	Problems 61–70	1.056	Problems 61–70	1.068
Problems 71–80	1.022	Problems 71–80	1.058	Problems 71–80	1.048
Problems 81–90	1.014	Problems 81–90	1.054	Problems 81–90	1.059
Problems 91–100	1.028	Problems 91–100	1.059	Problems 91–100	1.052

7 Conclusions

In this paper, we consider online scheduling on identical parallel machines with 2 GoS levels. When there are one 1-machine and more than one 2-machines, an online algorithm Alg1

is proposed with competitive ratio of at most 9/5. For the scheduling problem with more than one 1-machines and more than one 2-machines, we present an effective online algorithm Alg2 with competitive ratio of at most 2.280. When there are more than one 1-machines and one 2-machine used for processing jobs, we propose an online algorithm Alg3 with competitive ratio of at most 2. All the proposed algorithms satisfy that a 1-machine (2-machine) cannot be idle when there are an unprocessed 1-job (2-job), which can be seen as greedy algorithms. This property makes sure that the proposed algorithms perform well with the average ratios, which is very meaningful for real manufacturing.

As the complexity of the considered problem, it is difficult to find an online algorithm with competitive ratio of at most 2 for the second case. Future research is to find an effective online algorithm with a better competitive ratio for the second case of the considered problem.

References

[1] Mokotoff E, Parallel machine scheduling problems: A survey, *Asia-Pacific Journal of Operational Research*, 2001, **18**(2): 193–242.

[2] Franca P M, Gendreau M, Laporte G, et al., A composite heuristic for the identical parallel machine scheduling problem with minimum makespan objective, *Computers & operations research*, 1994, **21**(2): 205–210.

[3] Cheng R and Gen M, Parallel machine scheduling problems using memetic algorithms, *Computers & Industrial Engineering*, 1997, **33**(3–4): 761–764.

[4] Chen Z L and Powell W B, Solving parallel machine scheduling problems by column generation, *INFORMS Journal on Computing*, 1999, **11**(1): 78–94.

[5] Chang P C, Chen S H, and Lin K L, Two-phase sub population genetic algorithm for parallel machine-scheduling problem, *Expert Systems with Applications*, 2005, **29**(3): 705–712.

[6] Grisselle C and Armacost R L, Parallel machine scheduling with release time and machine eligibility restrictions, *Computers & Industrial Engineering*, 1997, **33**(1–2): 273–276.

[7] Liao L W and Sheen G J, Parallel machine scheduling with machine availability and eligibility constraints, *European Journal of Operational Research*, 2008, **184**(2): 458–467.

[8] Edis E B and Ozkarahan I, A combined integer/constraint programming approach to a resource-constrained parallel machine scheduling problem with machine eligibility restrictions, *Engineering Optimization*, 2011, **43**(2): 135–157.

[9] Tseng C T, Lee C H, Chiu Y S P, et al., A discrete electromagnetism-like mechanism for parallel machine scheduling under a grade of service provision, *International Journal of Production Research*, 2017, **55**(11): 3149–3163.

[10] Jongho P, Chang S Y, and Lee K, Online and semi-online scheduling of two machines under a grade of service provision, *Operations Research Letters*, 2006, **34**(6): 692–696.

[11] Jiang Y W, He Y, and Tang C M, Optimal online algorithms for scheduling on two identical machines under a grade of service, *Journal of Zhejiang University-Science*, 2006, **A7**(3): 309–314.

[12] Yong W, Ji M, and Yang Q F, Optimal semi-online scheduling algorithms on two parallel identical

machines under a grade of service provision, *International Journal of Production Economics*, 2012, **135**(1): 367–371.

[13] Zhang A, Jiang Y W, and Tan Z Y, Online parallel machines scheduling with two hierarchies, *Theoretical Computer Science*, 2009, **410**(38–40): 3597–3605.

[14] Tan Z Y and Zhang A, A note on hierarchical scheduling on two uniform machines, *Journal of Combinatorial Optimization*, 2010, **20**(1): 85–95.

[15] Jiang Y W, Online scheduling on parallel machines with two GoS levels, *Journal of Combinatorial Optimization*, 2008, **16**(1): 28–38.

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

[17] Marvin M and Shabtay Dvir, Scheduling unit length jobs on parallel machines with lookahead information, *Journal of Scheduling*, 2011, **14**(4): 335–350.

[18] Hong K S and Leung J Y T, On-line scheduling of real-time tasks, *IEEE Transactions on Computers*, 1992, **41**(10): 1326–1331.

[19] Chen B and Vestjens A, Scheduling on identical machines: How good is LPT in an online setting, *Operations Research Letters*, 1997, **21**(4): 165–169.

[20] John N and Seiden S S, An optimal online algorithm for scheduling two machines with release times, *Theoretical Computer Science*, 2001, **268**(1): 133–143.

[21] Grissele C and Armacost R L, Minimizing makespan on parallel machines with release time and machine eligibility restrictions, *International Journal of Production Research*, 2004, **42**(6): 1243–1256.

[22] Lee K, Joseph Y T L, and Pinedo M L, Scheduling jobs with equal processing times subject to machine eligibility constraints, *Journal of Scheduling*, 2011, **14**(1): 27–38.

[23] Xu J and Liu Z H, An optimal online algorithm for scheduling on two parallel machines with GoS eligibility constraints, *Journal of the Operations Research Society of China*, 2016, **4**(3): 371–377.

[24] Li S S and Zhang Y Z, On-line scheduling on parallel machines to minimize the makespan, *Journal of Systems Science and Complexity*, 2016, **29**(2): 472–477.

[25] Cai S, Liu A, and Liu K, Online scheduling of two identical machines under a grade of service provision, *Control Conference (CCC), 2017 36th Chinese IEEE*, 2017.

[26] Shmoys D B, Joel W, and David P W, Scheduling parallel machines on-line, *SIAM Journal on Computing*, 1995, **24**(6): 1313–1331.

[27] Ou J, Joseph Y T L, and Chung L L, Scheduling parallel machines with inclusive processing set restrictions, *Naval Research Logistics*, 2008, **55**(4): 328–338.

[28] Huo Y and Joseph Y T L, Fast approximation algorithms for job scheduling with processing set restrictions, *Theoretical Computer Science*, 2010, **411**(44–46): 3947–3955.