J. Oper. Res. Soc. China (2016) 4:371–377
DOI 10.1007/s40305-016-0119-1

An Optimal Online Algorithm for Scheduling on Two Parallel Machines with GoS Eligibility Constraints

Jia Xu[1] · **Zhao-Hui Liu**[1]

Received: 27 October 2015 / Revised: 5 January 2016 / Accepted: 21 February 2016 /
Published online: 5 March 2016

Abstract We consider the online scheduling problem on two parallel machines with the Grade of Service (GoS) eligibility constraints. The jobs arrive over time, and the objective is to minimize the makespan. We develop a $(1 + \alpha)$-competitive optimal algorithm, where $\alpha \approx 0.555$ is a solution of $\alpha^3 - 2\alpha^2 - \alpha + 1 = 0$.

Keywords Scheduling · Parallel machine · Eligibility constraint · Online algorithm

Mathematics Subject Classification 68M20 · 90B35

1 Introduction

We consider the following scheduling problem with machine eligibility constraints. There are n jobs $J_1, J_2, \cdots, J_n$ to be processed on m parallel machines $M_1, M_2, \cdots, M_m$. Every job J_j is associated with a release time r_j, a processing time p_j, and a processing set $\mathcal{M}_j \subseteq \{M_1, M_2, \cdots, M_m\}$, which mean that the job can only be processed at or after time r_j and on the machines in $\mathcal{M}_j$, and its processing takes p_j time units. The objective is to determine a schedule that minimizes the makespan $C_{\max}$, i.e., the maximum completion time of the jobs. We discuss the problem in the online

This research was supported by the National Natural Science Foundation of China (No. 11171106).

✉ Zhao-Hui Liu
zhliu@ecust.edu.cn

Jia Xu
xj851217@163.com

1 Department of Mathematics, East China University of Science and Technology, Shanghai 200237, China

setting. That is, the information of any job is available only after it is being released, even about its existence. But when a job appears, we have the option of scheduling it immediately or postponing its scheduling till some later time. In contrast, in the offline setting, we have full information about all jobs in advance. Using the 3-field notation of Graham et al. [1], we denote the online problem as $P|\mathcal{M}_j, r_j, \text{online}|C_{\max}$ and the corresponding offline problem as $P|\mathcal{M}_j, r_j|C_{\max}$. In this paper, we confine ourselves to a special type of processing set, i.e., the Grade of Service (GoS) processing set. In this case, for any two jobs J_i and J_j, it holds that either $\mathcal{M}_i \subseteq \mathcal{M}_j$ or $\mathcal{M}_i \supseteq \mathcal{M}_j$. Thus, the jobs and machines can be graded such that a job can be processed on a machine only when the grade of the job is not below the grade of the machine. We use $\mathcal{M}_j(\text{GoS})$ to indicate the special eligibility constraint. Also, we concern ourselves with the two-machine case, i.e., $P2|\mathcal{M}_j(\text{GoS}), r_j, \text{online}|C_{\max}$.

To evaluate the performance of an algorithm, we use the worst-case performance ratio and the competitive ratio for the offline problem and online problem, respectively. Let σ^* denote the offline optimal schedule and σ denote the schedule generated by the algorithm in context. Let $C_{\max}(\sigma^*)$ and $C_{\max}(\sigma)$ denote the makespan of σ^* and σ, respectively. If $C_{\max}(\sigma) \leqslant \rho C_{\max}(\sigma^*)$, this algorithm is said to be a ρ-approximation algorithm for the offline problem, and a ρ-competitive algorithm for the online problem.

When there are no eligibility constraints, i.e., each job can be processed on any machine, Chen and Vestjens [2] presented a 3/2-competitive algorithm for $P|r_j, \text{online}|C_{\max}$, and Noga and Seiden [3] showed a 1.382-competitive optimal algorithm for the two-machine problem $P2|r_j, \text{online}|C_{\max}$.

When there are some eligibility constraints, Shchepin and Vakhania [4] provided a $(2 - \frac{1}{m})$-approximation algorithm for the offline problem $P|\mathcal{M}_j|C_{\max}$ in which all jobs are available at time zero, and Muratore et al. [5] gave a Polynomial Time Approximation Scheme (PTAS) algorithm for the offline GoS constraints problem $P|\mathcal{M}_j(\text{GoS})|C_{\max}$. Shmoys et al. [6] showed that if there is a ρ-approximation algorithm for some scheduling problem in which all jobs are available at time zero, then there exists a 2ρ-competitive algorithm for the corresponding problem in which the jobs are released online over time. Therefore, $P|\mathcal{M}_j, r_j, \text{online}|C_{\max}$ has a $(4 - \frac{2}{m})$-competitive algorithm, and $P|\mathcal{M}_j(\text{GoS}), r_j, \text{online}|C_{\max}$ has a $(2 + \varepsilon)$-competitive algorithm. Xu and Liu [7] considered several problems with equal processing times and gave a $\sqrt{2}$-competitive optimal algorithm for $P|\mathcal{M}_j(\text{GoS}), p_j = p, r_j, \text{online}|C_{\max}$.

In this paper, we present a $(1+\alpha)$-competitive optimal algorithm for $P2|\mathcal{M}_j(\text{GoS}), r_j, \text{online}|C_{\max}$, where $\alpha \approx 0.555$ is a solution of $\alpha^3 - 2\alpha^2 - \alpha + 1 = 0$.

2 Algorithm

For $P2|\mathcal{M}_j(\text{GoS}), r_j, \text{online}|C_{\max}$, we need only to consider two types of processing sets: $\{M_1\}$ and $\{M_1, M_2\}$. For convenience, we call the jobs that can only be processed on M_1 1-jobs and the other jobs 2-jobs. Lee et. al. [8] have showed the following lemma.

Lemma 2.1 *Any online algorithm for* $P2|\mathcal{M}_j(\text{GoS}), r_j, \text{online}|C_{\max}$ *has a competitive ratio at least* $1 + \alpha \approx 1.555$, *where* α *is a solution of* $\alpha^3 - 2\alpha^2 - \alpha + 1 = 0$.

Here we present a $(1+\alpha)$-competitive algorithm H for $P2|\mathcal{M}_j(\text{GoS}), r_j, \text{online}|C_{\max}$. By Lemma 2.1, H is optimal. In Algorithm H, $J_{\max}(t)$ denotes the longest available 2-job at time t, and $p_{\max}(t)$ denotes its processing time.

Algorithm H

While M_1 is idle Do
  If there is an available 1-job, then schedule it on M_1;
  Else Do
    If there are two or more available 2-jobs, schedule the second longest 2-job on M_1;
    If there is only one available 2-job $J_{\max}(t)$ at current time t, and $p_{\max}(t) \leqslant \frac{\alpha}{1-\alpha} p_a$,
      where p_a is the processing time of the job J_a processed on M_2 at time t (assume $p_a = 0$ if M_2 is idle at this time), then schedule $J_{\max}(t)$ on M_1;
    Else keep M_1 idle until a new job is released.
While M_2 is idle Do
  If the current time $t \geqslant \alpha p_{\max}(t)$, then schedule $J_{\max}(t)$ on M_2;
  Else keep M_2 idle until time $t = \alpha p_{\max}(t)$.

Let σ and σ^* denote the schedule generated by Algorithm H and the offline optimal schedule, respectively. We use C_j and C_j^* to denote the completion times of job J_j in σ and σ^*, respectively, and use S_j and S_j^* to denote its start times in the two schedules. Let C and C^* denote the makespan of σ and σ^*, respectively. Let J_n be the last completed job, and L be the completion time of the machine that does not process J_n, in σ.

Obviously, if several 1-jobs $J_{u_1}, J_{u_2}, \cdots, J_{u_j}$ are scheduled continuously on M_1 in σ, then we can replace the jobs by a larger 1-job J_u with $p_u = \sum_{1 \leqslant i \leqslant j} J_{u_i}$ and $r_u = \min_{1 \leqslant i \leqslant j} r_{u_i}$. This replacement does not increase the length of σ^*, and keeps the length of σ and the positions of other jobs in σ unchanged. After this replacement, we can make sure there are no two 1-jobs scheduled on M_1 continuously in σ.

Lemma 2.2 *If J_j is scheduled on M_2 in σ, then $S_j \geqslant \alpha p_j$.*

Proof By Algorithm H, if J_j is scheduled on M_2, then J_j is just $J_{\max}(S_j)$. Therefore, $S_j \geqslant \alpha p_{\max}(S_j) = \alpha p_j$.

Lemma 2.3 *If J_n is 2-job, and $C - L > p_n$, then $C/C^* \leqslant 1+\alpha$.*

Proof If J_n is scheduled on M_1, by $C - L > p_n$, M_2 is idle at time S_n. Further, by the algorithm, we have $p_n \leqslant \frac{\alpha}{1-\alpha} p_a = 0$ and $r_n = S_n$; otherwise, J_n is scheduled on M_2. So, $C = r_n = C^*$. Next we suppose that J_n is scheduled on M_2.

If M_2 is idle just before J_n in σ, then $S_n = \alpha p_n$ or r_n, which means $C = (1+\alpha)p_n$ or $r_n + p_n$. Since $C^* \geqslant r_n + p_n$, we have $C/C^* \leqslant 1+\alpha$. Now we suppose $r_n < S_n$, and there is a job J_{n-1} finished at time S_n on M_2. As $C - L > p_n$, M_1 is idle at time S_n. Then, we have $p_n > \frac{\alpha}{1-\alpha} p_{n-1}$. Since we always schedule the current longest available 2-job on M_2, it holds that $r_n > S_{n-1}$. So, $C - C^* \leqslant p_{n-1}$. If $p_{n-1} \leqslant \frac{\alpha}{1+\alpha} C$, then $C - C^* \leqslant \frac{\alpha}{1+\alpha} C$, and the lemma holds. If $p_{n-1} > \frac{\alpha}{1+\alpha} C$, then $p_n > \frac{\alpha}{1-\alpha} p_{n-1} > \frac{\alpha^2}{1-\alpha^2} C$, and by Lemma 2.2, $S_{n-1} \geqslant \alpha p_{n-1} > \frac{\alpha^2}{1+\alpha} C$. So,

$$C = S_{n-1} + p_{n-1} + p_n > \frac{-\alpha^3 + \alpha^2 + \alpha}{1-\alpha^2} C.$$

Since $\alpha^3 - 2\alpha^2 - \alpha + 1 = 0$, we have $C > \frac{-\alpha^3+\alpha^2+\alpha}{1-\alpha^2} C = C$, a contradiction.

In the following analysis, we suppose that if J_n is 2-job, then $C - L \leqslant p_n$.

If $C - L \leqslant p_n$, then the machine that processes J_n is always busy from time L to C. If $C - L > p_n$, then J_n is 1-job, and M_2 is idle after time L. By the algorithm, no 2-jobs start on M_1 after time L. Thus, if M_1 has idle time in $[L, C]$, we can easily get $C = C^*$. In other words, we can also suppose that the machine which processes J_n is always busy from time L to C.

In the algorithm, M_1 can be idle for two reasons: There is no available job or there is only one available 2-job J_j with $p_j > \frac{\alpha}{1-\alpha} p_a$. Also, M_2 can be idle for two reasons: there is no available 2-job or the current time $t < \alpha p_{\max}(t)$. Call a time interval as idle time interval if there is at least one machine idle during this time interval. We can distinguish the idle time interval into two types:

(i) *a-type*: no available job for any idle machine during the interval;
(ii) *b-type*: there is an available job for some idle machine during the interval.

If there is no idle time before L, then by Lemma 2.2, M_2 does not process any job after time zero, and $L = 0$. Further, M_1 does not process any 2-job after time zero, i.e., σ is optimal. In the following, we suppose that there is at least one idle time interval before L in σ.

Let $[t_s, t_f]$ be the last idle time interval before L in σ. Notice that if there exists t' with $t_s < t' < t_f$ such that $[t_s, t']$ and $[t', t_f]$ are idle time interval of different types, we treat $[t', t_f]$ as the last idle time interval and let $t_s = t'$.

Let $\bar{J}(t_f)$ denote the set of jobs released before time t_f but completed after time t_f in σ. For $J_j \in \bar{J}(t_f)$, the processing mount after time t_f in σ and σ^* is equal to $\min\{p_j, C_j - t_f\}$ and $\min\{p_j, \max\{0, C_j^* - t_f\}\}$, respectively. Let

$$\delta_j = \max\{0, \min\{p_j, C_j - t_f\} - \min\{p_j, \max\{0, C_j^* - t_f\}\}\}.$$

Lemma 2.4 *If $\delta_j > 0$, then $\delta_j \leqslant S_j$ and $\delta_j \leqslant t_f - S_j^*$ hold.*

Proof It follows from $\delta_j > 0$ that $\min\{p_j, C_j - t_f\} > \min\{p_j, \max\{0, C_j^* - t_f\}\}$. Then, $p_j > C_j^* - t_f = S_j^* + p_j - t_f$, i.e., $t_f - S_j^* > 0$.

When $C_j^* < t_f$, we have $\delta_j = \min\{p_j, C_j - t_f\}$ and $S_j^* + p_j = C_j^* < t_f$. Then, $\delta_j \leqslant p_j < t_f - S_j^*$ and $\delta_j \leqslant C_j - t_f < C_j - (S_j^* + p_j) = S_j - S_j^* \leqslant S_j$. When $t_f \leqslant C_j^* < p_j + t_f$, $\delta_j = \min\{p_j, C_j - t_f\} - (C_j^* - t_f)$. Thus, $\delta_j \leqslant (C_j - t_f) - (C_j^* - t_f) = S_j - S_j^* \leqslant S_j$ and $\delta_j \leqslant p_j - (C_j^* - t_f) = p_j - (S_j^* + p_j - t_f) = t_f - S_j^*$.

Let $\delta = \sum_{J_j \in \bar{J}(t_f)} \delta_j$. If $\bar{J}(t_f) = \varnothing$, let $\delta = 0$.

Lemma 2.5 *If $[t_s, t_f]$ is a-type, then $\delta \leqslant t_f$.*

Proof If M_1 is idle during $[t_s, t_f]$, then all the jobs scheduled at or after time t_f must be released at or after time t_f. So, $|\bar{J}(t_f)| \leqslant 1$. If $\bar{J}(t_f) = \varnothing$, the lemma holds. If $|\bar{J}(t_f)| = 1$, say $\bar{J}(t_f) = \{J_b\}$, then according to Lemma 2.4, $\delta = \delta_b \leqslant S_b \leqslant t_f$.

If M_2 is idle during $[t_s, t_f]$, then all the 2-jobs scheduled at or after time t_f must be released at or after time t_f. If there is no job scheduled before t_f and completed at or after time t_f on M_1, then all the jobs scheduled at or after time t_f must be released at or after time t_f. Thus, $\bar{J}(t_f) = \emptyset$, and the lemma holds. Now suppose there is a job J_b scheduled before time t_f and completed at or after time t_f on M_1. If all the jobs scheduled after J_b on M_1 are released at or after time t_f, then $\bar{J}(t_f) = \{J_b\}$ and $\delta = \delta_b \leqslant S_b \leqslant t_f$. If there exists another job J_c in $\bar{J}(t_f)$, then J_c must be 1-job and $r_c > S_b$. Thus, J_b is 2-job. If $\delta_c = 0$, then $\delta = \delta_b \leqslant t_f$. If $\delta_c > 0$, since $S_c^* \geqslant r_c > S_b$, we have $\delta_c \leqslant t_f - S_c^* \leqslant t_f - S_b$. Thus, $t_f \geqslant \delta_c + S_b \geqslant \delta_c + \delta_b = \delta$.

Theorem 2.6 *If* $[t_s, t_f]$ *is a-type, then* $C/C^* \leqslant 1 + \alpha$.

Proof First suppose that J_n is 2-job. If $S_n < t_f$, then $[t_s, t_f]$ is merely an a-type idle time interval on the machine that does not process J_n in σ. Therefore, all the jobs scheduled at or after time t_f must be released at or after t_f, and deleting them does not change the positions of the other jobs (including J_n). But the deletion operation will decrease L and cause $L \leqslant t_s$, and hence, we turn to deal with the new σ. Now suppose $S_n \geqslant t_f$. Since J_n is 2-job, we have $C - L \leqslant p_n$. Further, since $[t_s, t_f]$ is an a-type idle time interval, we have $r_n \geqslant t_f$, and by Lemma 2.5, we have $\delta \leqslant t_f$. Thus,

$$C^* \geqslant r_n + p_n \geqslant t_f + p_n \geqslant \delta + C - L.$$

By the definition of δ, for those jobs released before t_f and completed at or after t_f in σ, the processing mount after t_f in σ^* is δ less than in σ. Then we have

$$C^* \geqslant t_f + \frac{1}{2}(C + L - 2t_f - \delta) = C - \frac{1}{2}(C - L + \delta) \geqslant C - \frac{1}{2}C^*.$$

It leads to $C \leqslant \frac{3}{2}C^*$.

Next we suppose that J_n is 1-job. If no job is completed at time S_n on M_1, then $r_n = S_n$ and σ is optimal. If there is some job J_{n-1} completed at time S_n on M_1, then J_{n-1} is 2-job. As in the case where J_n is 2-job, we can suppose $S_{n-1} \geqslant t_f$. Notice that $r_n > S_{n-1}$. If $p_{n-1} \leqslant \frac{\alpha}{1+\alpha}C$, we have $C - C^* < p_{n-1} \leqslant \frac{\alpha}{1+\alpha}C$, which leads to $C < (1+\alpha)C^*$. If $p_{n-1} > \frac{\alpha}{1+\alpha}C$, as J_{n-1} is a 2-job scheduled on M_1, there must be some job J_k on M_2 such that $p_k \geqslant \frac{1-\alpha}{\alpha} p_{n-1} > \frac{1-\alpha}{1+\alpha}C$. Clearly, $C_k > S_{n-1} \geqslant t_f$. If $S_k < t_f$, then $\bar{J}(t_f) = \{J_k\}$ and $\delta = \delta_k \leqslant S_k$, which implies $L - \delta \geqslant p_k > \frac{1-\alpha}{1+\alpha}C$. Then,

$$C^* \geqslant t_f + \frac{1}{2}(C + L - 2t_f - \delta) = \frac{1}{2}C + \frac{1}{2}(L - \delta) > \frac{1}{1+\alpha}C.$$

If $S_k \geqslant t_f$, then $L - t_f \geqslant L - S_k \geqslant p_k$. By Lemma 2.5, $\delta \leqslant t_f$. Then we have

$$C^* \geqslant t_f + \frac{1}{2}(C + L - 2t_f - \delta) \geqslant \frac{1}{2}C + \frac{1}{2}(L - t_f) > \frac{1}{1+\alpha}C.$$

This completes the proof.

In the remaining part of this section, we suppose that the last idle time interval $[t_s, t_f]$ is b-type. The following Lemmas 2.7 and 2.8 are obvious for Algorithm H.

Lemma 2.7 *If M_1 is idle during $[t_1, t_2]$, then there is at most one job released before time t_2 and scheduled at or after time t_2.*

In fact, when $[t_1, t_2]$ is a b-type idle time interval, there is only one job released before time t_2 and scheduled at or after time t_2; when $[t_1, t_2]$ is an a-type idle time interval, there is no job released before time t_2 and scheduled at or after time t_2.

Lemma 2.8 *If M_2 is idle during $[t_s, t_f]$ which is b-type, then job $J_{\max}(t_f)$ is scheduled at time t_f on M_2, where $t_f = \alpha p_{\max}(t_f)$.*

Theorem 2.9 *If $[t_s, t_f]$ is b-type, then $C/C^* \leqslant 1 + \alpha$.*

Proof We first consider the case where M_2 is idle during $[t_s, t_f]$. If $J_{\max}(t_f)$ is completed later than L, then $J_{\max}(t_f)$ is the last completed job and the theorem holds obviously. So, we suppose that $J_{\max}(t_f)$ is completed no later than L in σ. Let $[t'_s, t'_f]$ be the last idle time interval on M_1 (if there is not such time interval, we let $t'_s = t'_f = 0$). Clearly, $t'_f \leqslant t_f$. By Lemma 2.7, there is at most one job, say J_k, released before t'_f and scheduled at or after time t'_f in σ. Let δ'_k denote the processing mount of J_k before time t'_f in σ^*. Clearly, $\delta'_k \leqslant t'_f$. Then we have

$$C^* \geqslant \frac{1}{2}(C + L - t_f - t'_f - \delta'_k) + t'_f \geqslant \frac{1}{2}(C + L - t_f).$$

If $L - t_f \geqslant \frac{1-\alpha}{1+\alpha}C$, then $C^* \geqslant \frac{1}{1+\alpha}C$, and the theorem holds. So, we suppose $L - t_f < \frac{1-\alpha}{1+\alpha}C$. By Lemma 2.8, $t_f = \alpha p_{\max}(t_f) \leqslant \alpha(L - t_f)$. Then $L = t_f + L - t_f < (1-\alpha)C$.

If J_n is 2-job, then

$$p_n \geqslant C - L > \alpha C > \frac{1-\alpha}{1+\alpha}C > L - t_f \geqslant p_{\max}(t_f).$$

Thus, $r_n > t_f$, and $C^* \geqslant r_n + p_n > t_f + C - L$, and $C - C^* < L - t_f < \frac{1-\alpha}{1+\alpha}C$, which leads to $C < \frac{1+\alpha}{2\alpha}C^* < (1+\alpha)C^*$.

If J_n is 1-job, as in the proof of Theorem 2.6, we can find the last 2-job J_{n-1} on M_1. By the algorithm, there is a job J_k, with $p_k \geqslant \frac{1-\alpha}{\alpha}p_{n-1}$, scheduled on M_2. If $S_k \geqslant t_f$, then $L - t_f \geqslant p_k \geqslant \frac{1-\alpha}{\alpha}p_{n-1}$. If $S_k < t_f$, we have $p_k \leqslant \frac{S_k}{\alpha} < \frac{t_f}{\alpha} = p_{\max}(t_f)$. Again, $L - t_f \geqslant p_{\max}(t_f) > p_k \geqslant \frac{1-\alpha}{\alpha}p_{n-1}$. So, $p_{n-1} \leqslant \frac{\alpha}{1-\alpha}(L - t_f) < \frac{\alpha}{1+\alpha}C$. Noticing that $r_n > S_{n-1}$, we have $C - C^* < p_{n-1} < \frac{\alpha}{1+\alpha}C$. Consequently, $C/C^* \leqslant 1 + \alpha$ holds.

Next we consider the case where M_1 is idle during $[t_s, t_f]$. If M_2 is idle at time t_f, then the same argumentation as above works. Thus, we suppose there is a job J_a processed on M_2 at time t_f. By Lemma 2.7, there is only one job released before time t_f and scheduled at or after time t_f. Clearly, the only job must be longer than $\frac{\alpha}{1-\alpha}p_a$, so it is released after S_a. That is, all the jobs scheduled at or after time t_f are released after time S_a. Let δ'_a denote the processing mount of J_a before time S_a in σ^*. Clearly, $\delta'_a \leqslant S_a$. Then we have

$$C^* \geqslant \frac{1}{2}(C + L - t_f - S_a - \delta'_a) + S_a \geqslant \frac{1}{2}(C + L - t_f). \tag{2.1}$$

If J_n is 1-job, then just as before, we can find the last 2-job J_{n-1} on M_1. Notice that $r_n > S_{n-1}$. If $p_{n-1} \leqslant \frac{\alpha}{1+\alpha}C$, we have $C - C^* < p_{n-1} \leqslant \frac{\alpha}{1+\alpha}C$ and $C < (1+\alpha)C^*$. If $p_{n-1} > \frac{\alpha}{1+\alpha}C$, then there is a job J_k scheduled at or after time S_a on M_2 such that $p_k \geqslant \frac{1-\alpha}{\alpha}p_{n-1} > \frac{1-\alpha}{1+\alpha}C$. If J_a and J_k are different jobs, then J_k is scheduled after J_a, and we have $L - t_f \geqslant p_k > \frac{1-\alpha}{1+\alpha}C$. Notice that there is a 2-job longer than $\frac{\alpha}{1-\alpha}p_a$ released before t_f and scheduled at or after time t_f. This 2-job is scheduled after J_a in σ. Thus, if J_a and J_k are the same job, we have $L - t_f \geqslant \frac{\alpha}{1-\alpha}p_a > p_k > \frac{1-\alpha}{1+\alpha}C$. It follows from (2.1) that $C/C^* \leqslant 1 + \alpha$.

If J_n is 2-job, then $p_n \geqslant C - L$. By (2.1), we need only to consider the case of $L - t_f < \frac{1-\alpha}{1+\alpha}C$. If $r_n \geqslant t_f$, then

$$C^* \geqslant r_n + p_n \geqslant t_f + (C - L) > \frac{2\alpha}{1+\alpha}C > \frac{1}{1+\alpha}C.$$

Thus, we suppose that J_n is just the job released before t_f and scheduled at or after t_f. Clearly, $p_n > \frac{\alpha}{1-\alpha}p_a$ and $S_n \geqslant C_a$. If $S_n > C_a$, then there exists a job J_j scheduled on M_2 such that $p_j \geqslant p_n$ and $r_j \geqslant t_f$. Thus, $C^* \geqslant r_j + p_j \geqslant t_f + p_n > \frac{2\alpha}{1+\alpha}C$, and the theorem holds. If $S_n = C_a$, then

$$C = S_a + p_a + p_n > \alpha p_a + p_a + \frac{\alpha}{1-\alpha}p_a = \frac{1+\alpha-\alpha^2}{1-\alpha}p_a = \frac{1+\alpha}{\alpha}p_a,$$

where the last equality follows from $\alpha^3 - 2\alpha^2 - \alpha + 1 = 0$. As $p_n > p_a$, we have $r_n > S_a$. Then, $C - C^* \leqslant p_a < \frac{\alpha}{1+\alpha}C$, and the theorem holds.

Combining the above analysis with Lemma 2.1, we obtain the following result.

Theorem 2.10 *Algorithm H is a* $(1+\alpha)$*-competitive optimal algorithm for* $P2|\mathcal{M}_j$ (GoS), r_j, online$|C_{\max}$.

References

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

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

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

[4] Shchepin, E.V., Vakhania, N.: An optimal rounding gives a better approximation for scheduling unrelated machines. Oper. Res. Lett. **33**, 127–133 (2005)

[5] Muratore, G., Schwarz, U.M., Woeginger, G.J.: Parallel machine scheduling with nested job assignment restrictions. Oper. Res. Lett. **38**, 47–50 (2010)

[6] Shmoys, D.B., Wein, J., Williamson, D.P.: Scheduling parallel machines on-line. SIAM J. Comput. **24**(6), 1313–1331 (1995)

[7] Xu, J., Liu, Z.: Online scheduling with equal processing times and machine eligibility constraints. Theor. Comput. Sci. **572**, 58–65 (2015)

[8] Lee, K., Leung, J.Y.-T., Pinedo, M.L.: Makespan minimization in online scheduling with machine eligibility. 4OR **8**, 331–364 (2010)