J Comb Optim (2015) 29:781–795
DOI 10.1007/s10878-013-9627-7

Optimal online algorithms on two hierarchical machines with tightly-grouped processing times

An Zhang · Yiwei Jiang · Lidan Fan · Jueliang Hu

Published online: 15 May 2013

Abstract This paper considers an online hierarchical scheduling problem on two parallel identical machines. The objective is to minimize the makspan. It is assumed that all jobs have bounded processing times in between p and rp, where $p > 0$ and $r \geq 1$. We first improve a previous result by giving an optimal online algorithm for the non-preemptive version. For the preemptive version, we present an optimal preemptive algorithm without introducing idle time for all $r \geq 1$. If the algorithm is allowed to use idle time, we show that the semi-online information that jobs are tightly-grouped cannot help improve the bound of the pure online problem.

Keywords Online scheduling · Hierarchy · Competitive ratio

A. Zhang
Institute of Operational Research and Cybernetics, Hangzhou Dianzi University, Hangzhou 310018, People's Republic of China
e-mail: anzhang@hdu.edu.cn

Y. Jiang (✉)
Department of Mathematics, Zhejiang Sci-Tech University, Hangzhou 310018, People's Republic of China
e-mail: ywjiang@zstu.edu.cn

L. Fan
Department of Computer Science, The University of Texas at Dallas, Dallas, TX 75080, USA
e-mail: ldfan28@gmail.com

J. Hu
School of Science, Zhejiang Sci-Tech University, Hangzhou 310018, People's Republic of China
e-mail: hujlhz@163.com

1 Introduction

In this paper, we study online hierarchical scheduling on two parallel identical machines where all jobs have their processing times in between p and rp ($p > 0, r \geq 1$). Without loss of generality, we assume that $p = 1$ and it is allowed that the jobs with processing times p or rp may not come. Jobs arrive one by one in a sequence $J_1, J_2, \ldots, J_n$ and each job has to be processed on a machine before the next job arrives. The job J_j has a processing time (or size) $p_j > 0$ and a hierarchy $g_j = 1$ or 2, $j = 1, \ldots, n$. For simplicity, we identify jobs with their processing times. The machine M_i, $i = 1, 2$ has a hierarchy $g(M_i) = i$ associated with it. M_i can process J_j only when $g(M_i) \leq g_j$. We define the *load* of a machine as the completion time of the machine, i.e., the total processing times of the jobs processed on it. The objective is to minimize the maximum load of the machines or the so-called makespan. We consider both the non-preemptive and preemptive version version, where preemption indicates that any job can be split between its permitted machines without overlap.

The performance of an algorithm A for online problem is often evaluated by its competitive ratio, which is defined as the smallest number ρ such that for any job sequence $\mathcal{J}$, $C^A(\mathcal{J}) \leq \rho C^*(\mathcal{J})$, where $C^A(\mathcal{J})$ (or in short C^A) denotes the makespan produced by A and $C^*(\mathcal{J})$ (or in short C^*) denotes the optimal makespan in an offline version. In online scheduling, it is often possible to prove that there is a lower bound to the competitive ratio achievable by any (deterministic) online algorithm. In such cases, an online algorithm is said to be *optimal* if its competitive ratio can be shown to be equal to the lower bound.

For online hierarchical scheduling on m parallel identical machines with general hierarchy settings, Bar-Noy et al. (2001) firstly designed an $e + 1 \approx 3.718$-competitive algorithm for the non-preemptive version (this problem was also considered in Crescenzi et al. 2004). A recent study on this problem can be seen in Tan and Zhang (2011). For $m = 2$, Jiang et al. (2006) and Park et al. (2006) independently presented an optimal online algorithm, both of which has a competitive ratio of $5/3$. Afterwards, Jiang (2008) studied another extended case with m identical machines and exactly two hierarchies. He showed a lower bound of 2 and presented an online algorithm with competitive ratio $\frac{12+4\sqrt{2}}{7} \approx 2.522$. Recently, Zhang et al. (2009) improved the result to $1 + \frac{m^2-m}{m^2-km+k^2} < 7/3$, where k is the number of machines with high hierarchy.

Jiang et al. (2006) studied the preemptive version in which idle time is not allowed and they presented an optimal algorithm with a competitive ratio of $3/2$ on two identical machines. In Park et al. (2006), a semi-online model with known total processing time was considered and an optimal algorithm was presented with competitive ratio of $3/2$ on two identical machines. Wu et al. (2012) studied another two semi-online models, where the optimal offline value or the largest processing time was known in advance, optimal algorithms were presented for both models on two identical machines. In Chassid and Epstein (2008), Dósa and Epstein (2008), Tan and Zhang (2010), the authors studied the problems of hierarchial scheduling on two uniform machines and presented optimal (non-)preemptive algorithms for online and semi-online models.

In this paper, we consider a hierarchical scheduling problem, where a semi-online information that job processing times are tightly-grouped is involved ($p_j \in [1, r]$ for all J_j). Such a semi-online model was first introduced by He and Zhang (1999) in the problem with no hierarchy constraint. Liu et al. (2011) studied this model in hierarchical scheduling on two identical machines. They proved a lower bound of

$$\begin{cases} \frac{r+1}{2}, & 1 \le r < 2, \\ \frac{3}{2}, & 2 \le r < 5, \\ \frac{r+4}{6}, & 5 \le r < 6. \\ \frac{5}{3}, & r \ge 6. \end{cases}$$

However, the algorithm presented in their paper is only shown to be optimal only in the case where $r \ge \frac{25}{14}$ and $C^* \ge 20r$. This paper improves the lower bound to

$$\alpha(r) = \begin{cases} \frac{2r+1}{r+1}, & 1 \le r < \phi, \\ \phi, & \phi \le r < 3\phi - 2, \\ \frac{r+2}{3}, & 3\phi - 2 \le r < 3, \\ \frac{5}{3}, & r \ge 3, \end{cases}$$

where $\phi = \frac{1+\sqrt{5}}{2}$, and proposes a new algorithm which is optimal for any $r \ge 1$.

In addition, we show that the semi-online information cannot help improve the bound of the pure online problem if the preemptive algorithms is allowed to use idle time. Then we only need to consider the preemptive algorithm without introducing idle time. We present an optimal algorithm with a competitive ratio of

$$\beta(r) = \begin{cases} \frac{2r+2}{r+2}, & 1 \le r < 2, \\ \frac{3}{2}, & r \ge 2. \end{cases}$$

The rest of the paper is organized as follows. Section 2 introduces the preliminaries and lower bounds. Sections 3 and 4 consider the online algorithms for the non-preemptive version and the preemptive version, respectively. Finally, some conclusions are made in Sect. 5.

2 Preliminaries and lower bounds

The following notations and definitions are used in the remainder of the paper. Let T_j and T_{ji} be the total processing times of the first j jobs and the total processing times of the jobs with hierarchy i in the first j jobs, respectively. Let $p_j^{\max}$ be the largest processing time of the first j jobs.

Let

$$LB_j = \max\left\{p_j^{\max}, \frac{T_j}{2}, T_{j1}\right\}, \quad j = 1, \ldots, n. \tag{1}$$

Clearly, LB_j is nondecreasing with respect to j. Denote by L_j^* the optimal makespan of the first j jobs, thus $C^* = L_n^*$. The following lemma shows that LB_j is a lower bound on L_j^*.

Lemma 2.1 (Zhang et al. 2009; Dósa and Epstein 2008) *$L_j^* \geq LB_j$, for any $j \geq 1$. Further, $L_j^* = LB_j$ if preemption is allowed.*

Note that the functions $\alpha(r)$ and $\beta(r)$ are monotonically increasing. To simplify the presentation, we will drop the dependence on r and always write α (or β) instead of $\alpha(r)$ (or $\beta(r)$).

In the following, we give the lower bounds of the considered problems.

Theorem 2.2 *For the non-preemptive version, no online algorithm exists with competitive ratio strictly smaller than α.*

Proof We use the adversary method to establish the result. Assume that there exists an online non-preemptive algorithm A with competitive ratio C.

(a) $1 \leq r < \phi$. The first job with $p_1 = 1$ and $g_1 = 2$ arrives. It is clear that the algorithm A should schedule it on M_2, since otherwise, the second and the last job with $p_2 = 1$ and $g_2 = 1$ arrives and has to be assigned to M_1, which implies that $C \geq 2 \geq \frac{2r+1}{r+1}$. Therefore we assume that the algorithm schedules p_1 on M_2. Then the second job with $p_2 = r$ and $g_2 = 2$ arrives. If it is also scheduled on M_2 by the algorithm, then no more job arrives. Thus the makespan generated by algorithm A is $r + 1$ while the optimal makespan equals r, which implies that $C \geq \frac{1+r}{r}$. On the other hand, if the second job is scheduled on M_1, then the last two jobs with $p_3 = 1$, $p_4 = r$ and $g_3 = g_4 = 1$ arrive. Clearly, the makespan generated by the algorithm is $2r + 1$ and the optimal makespan is $r + 1$, implying that $C \geq \frac{2r+1}{r+1}$. Thus we have $C \geq \min\{\frac{1+r}{r}, \frac{2r+1}{r+1}\} \geq \frac{2r+1}{r+1}$ due to $r < \phi$.

(b) $\phi \leq r < 3\phi - 2$. By replacing the jobs of size r with jobs of size ϕ in (a), we can obtain $C \geq \min\{\frac{1+\phi}{\phi}, \frac{2\phi+1}{\phi+1}\} = \phi$.

(c) $3\phi - 2 \leq r < 3$. Let the first two jobs be $p_1 = p_2 = 1$ with $g_1 = g_2 = 2$. Clearly, they have to be scheduled on different machines to avoid that $C \geq 2$. Then the third job with $p_3 = 1$ and $g_3 = 2$ arrives. If it is scheduled on M_1, then there comes the last job with $p_4 = r$ and $g_4 = 1$. Thus the makespan produced by A and optimal makespan should be $r + 2$ and 3 respectively, which follows that $C \geq \frac{r+2}{3}$. On the other hand, if the third job is scheduled on M_2, then the fourth job with $p_4 = r$ and $g_4 = 2$ arrives. Similarly, it should be scheduled on M_1 to avoid that $C \geq \frac{r+2}{3}$. Then come the last two jobs, both of which are of hierarchy 1 and have processing time r. It implies that the makespan produced by A and the optimal schedule are $3r + 1$ and $r + 3$, respectively. Hence we have $C \geq \frac{3r+1}{r+3} \geq \frac{r+2}{3}$ due to $r < 3$.

(d) $r \geq 3$. The desired result can be obtained by using the instances in (c) with $r = 3$.

□

In (Jiang et al. 2006; Park et al. 2006), online algorithms with competitive ratio $\frac{5}{3}$ were presented and shown to be optimal for the problem with arbitrary processing

times. Clearly the algorithm still works for our problem with the same competitive ratio. By Theorem 2.2, it remains optimal when $r \geq 3$. However, when $1 \leq r < 3$, we have to design a more competitive algorithm. If preemption is allowed, then the following result (see also in Dósa and Epstein 2008) shows that the semi-online information is of no value if the algorithm uses idle time.

Theorem 2.3 *For the preemptive version where idle time is allowed, no online algorithm exists with competitive ratio strictly smaller than $\frac{4}{3}$, even when all jobs have the same size.*

Proof Let the first two jobs be of size 1 and of hierarchy 2. If the algorithm assigns a total size of at least $\frac{4}{3}$ to M_2, then the sequence of jobs terminates, resulting $C^A \geq \frac{4}{3}$. Hence, $\frac{C^A}{C^*} \geq \frac{4}{3}$. If this is not true, then at least $\frac{2}{3}$ must be assigned to M_1. Now the last two jobs with the same size 1 and the same hierarchy 1 arrive. Clearly we have $C^A \geq 2 + \frac{2}{3} = \frac{8}{3}$ and $C^* = 2$, thus $\frac{C^A}{C^*} \geq \frac{4}{3}$. □

When idle time is allowed, Dósa and Epstein have proposed a $\frac{4}{3}$-competitive algorithm for the pure online problem where the processing times of jobs are arbitrary (Dósa and Epstein 2008), hence the same algorithm can be applied to our problem, which is also best possible. In other words, the semi-online information that jobs are tightly-grouped is not useful. However, we find it is valuable if the preemptive algorithm is not allowed to use idle time.

Theorem 2.4 *For the preemptive version where idle time is not allowed, no online algorithm exists with competitive ratio strictly smaller than β.*

Proof Note when $r \geq 2$, Jiang et al. (2006) have proved a lower bound of $\frac{3}{2}$ for preemptive algorithms without introducing idle times, thus we only need to consider the case of $1 \leq r \leq 2$. The first job with $p_1 = 1$ and $g_1 = 2$ should be assigned completely to a machine since splitting it would introduce an idle time, which is not allowed in our problem. If it is assigned to machine M_1, then the second and last job with $p_2 = 1$ and $g_2 = 1$ arrives. Since p_2 has to be assigned to machine M_1 due to $g_2 = 1$, it follows that the makespan produced by the algorithm is 2, while the optimal makespan is 1. Hence we have $\frac{C^A}{C^*} \geq 2 > \frac{2r+2}{r+2}$. On the other hand, if p_1 is assigned to machine M_2, then the second job with $p_2 = 1$ and $g_2 = 1$ arrives, which has to be assigned to machine M_1. Now both loads of two machines are 1. Then the third and last job with $p_3 = r$ and $g_3 = 2$ arrives. Thus the makespan is at least $1 + r$. As the optimal makespan is $\frac{r+2}{2}$ by Lemma 2.1, we can conclude $\frac{C^A}{C^*} \geq \frac{2r+2}{r+2}$. □

3 An optimal non-preemptive algorithm

In this section, we consider the non-preemptive problem. Unlike in Park et al. (2006), we have to know the accurate optimal objective values in the algorithm. This is perhaps due to the tightly-grouped property of the jobs. Since only the values of the first several jobs are calculated, this can be done in a constant time. Denote

$$\overline{LB}_j = \begin{cases} L_j^*, & 1 \leq j \leq 5, \\ LB_j, & j > 5, \end{cases}$$

then $L_j^* \geq \overline{LB}_j \geq LB_j$ by Lemma 2.1. To describe the algorithm easily, let L_j^i be the completion time of machine M_i at the moment right after the j-th job J_j is scheduled.

Algorithm A

1. Initially set $L_0^i = 0$ for $i = 1, 2$ and let $j = 1$.
2. If $g_j = 1$, schedule p_j on M_1.
3. If $g_j = 2$, and $L_{j-1}^2 + p_j \leq \alpha \overline{LB}_j$, schedule p_j on M_2. Otherwise, schedule p_j on M_1.
4. If no new job arrives, stop. Otherwise, $j = j + 1$, return to step 2.

Now we begin to show the competitive ratio of algorithm A. Without loss of generality, suppose the last job p_n determines the makespan of A.

Theorem 3.1 *If $g_n = 2$, then $C^A/C^* \leq \alpha$.*

Proof If p_n is processed on M_2 by A, then it is clear that $C^A = L_{n-1}^2 + p_n \leq \alpha \overline{LB}_n \leq \alpha C^*$ by step 3. On the other hand, we consider the case that p_n is processed on M_1. By step 3, we have $L_{n-1}^2 + p_n > \alpha \overline{LB}_n$, which, together with Lemma 2.1, implies that

$$\overline{LB}_n \geq LB_n \geq \frac{L_{n-1}^2 + L_{n-1}^1 + p_n}{2} > \frac{\alpha \overline{LB}_n + L_{n-1}^1}{2}.$$

It follows that $L_{n-1}^1 \leq (2-\alpha)\overline{LB}_n$. Hence

$$C^A = L_{n-1}^1 + p_n \leq (2-\alpha)\overline{LB}_n + C^* \leq (2-\alpha)C^* + C^* = (3-\alpha)C^* \leq \alpha C^*,$$

where the last inequality is due to $\alpha \geq \frac{3}{2}$. □

In the following, it suffices to consider the case that $g_n = 1$ and clearly p_n is scheduled on M_1, i.e., $C^A = L_n^1$. If all jobs processed on M_1 by the Algorithm A are of hierarchy 1, then A has already generated an optimal schedule and thus the desired result holds trivially. In view of this, we consider the instances in which there exist jobs with hierarchy 2 scheduled on M_1. We denote by p_j the last one among these jobs, that is, all the subsequent jobs scheduled on M_1 are of hierarchy 1 after p_j is assigned. Therefore,

$$T_{n1} \geq L_n^1 - L_j^1. \tag{2}$$

From step 3 of the Algorithm A, we know that at least one job is scheduled on M_2 before the arrival of p_j. And since J_j is assigned to M_1, we have

$$L_{j-1}^2 + p_j > \alpha \overline{LB}_j \tag{3}$$

by the algorithm's rule. Moreover, we have the following observation.

Lemma 3.2 *Before the arrival of* p_j, $L^1_{j-1} < L^2_{j-1}$.

Proof By (3), we obtain $L^2_{j-1} + p_j > \alpha \overline{LB}_j \geq \frac{3}{2} LB_j$. If $L^1_{j-1} \geq L^2_{j-1}$, we have

$$\frac{L^2_{j-1} + p_j}{LB_j} \leq \frac{L^2_{j-1} + p_j}{\max\left\{p_j, \frac{L^2_{j-1}+L^1_{j-1}+p_j}{2}\right\}} \leq \frac{L^2_{j-1} + p_j}{\max\left\{p_j, \frac{2L^2_{j-1}+p_j}{2}\right\}}$$
$$= \begin{cases} \frac{L^2_{j-1}+p_j}{p_j}, & p_j \geq 2L^2_{j-1} \\ \frac{2(L^2_{j-1}+p_j)}{2L^2_{j-1}+p_j}, & p_j < 2L^2_{j-1} \end{cases} \leq \frac{3}{2},$$

which is a contradiction. Hence we must have $L^1_{j-1} < L^2_{j-1}$. □

Let

$$\lambda = \begin{cases} r, & 1 \leq r < \phi, \\ \phi, & \phi \leq r < 3\phi - 2, \\ \frac{r-1}{4-r}, & 3\phi - 2 \leq r < 3. \end{cases}$$

Note that $\lambda = \frac{\alpha-1}{2-\alpha} \geq 1$ and $\alpha = \frac{2\lambda+1}{\lambda+1}$. We now give an important lemma below.

Lemma 3.3 *If* $L^1_j \leq \lambda L^2_{j-1}$, *then* $C^A/C^* \leq \alpha$.

Proof By (1), (2) and Lemma 2.1, we have

$$C^* \geq LB_n \geq \max\left\{\frac{T_n}{2}, T_{n1}\right\} \geq \max\left\{\frac{L^1_n + L^2_{j-1}}{2}, L^1_n - L^1_j\right\}.$$

Hence, together with $L^1_j \leq \lambda L^2_{j-1}$, we get

$$\frac{C^A}{C^*} \leq \frac{L^1_n}{\max\left\{\frac{L^1_n+L^2_{j-1}}{2}, L^1_n - L^1_j\right\}}$$
$$= \begin{cases} \frac{L^1_n}{L^1_n - L^1_j}, & \text{if} \quad L^1_j + L^2_{j-1} \leq L^1_n - L^1_j, \\ \frac{2L^1_n}{L^1_n + L^2_{j-1}}, & \text{if} \quad L^1_j + L^2_{j-1} > L^1_n - L^1_j, \end{cases}$$
$$\leq \frac{(L^1_j + L^2_{j-1}) + L^1_j}{L^1_j + L^2_{j-1}} \leq \frac{2\lambda+1}{\lambda+1} = \alpha.$$

□

From Lemma 3.3, we only need to show in the remainder of this section that $L^1_j \leq \lambda L^2_{j-1}$ must hold. Furthermore, we can assume that

$$L^1_j = L^1_{j-1} + p_j > L^2_{j-1}, \tag{4}$$

because the desired result holds if $L_j^1 \le L_{j-1}^2$ by Lemma 3.3.

Lemma 3.4 *If $1 \le r < \phi$, then $L_j^1 \le \lambda L_{j-1}^2$.*

Proof Since $1 \le r < \phi$, we know $\alpha = \frac{2r+1}{r+1}$ and $\lambda = r$.

Case 1. At least two jobs are scheduled on M_2 before the arrival of p_j. Hence $L_{j-1}^2 \ge 2$. By (3), we have

$$L_{j-1}^2 + p_j > \frac{2r+1}{r+1} \cdot \overline{LB}_j \ge \frac{2r+1}{r+1} \cdot \frac{L_{j-1}^2 + L_j^1}{2},$$

i.e., $(2r+1)L_j^1 < L_{j-1}^2 + 2(r+1)p_j$. Combining it with $p_j \le r$ and $L_{j-1}^2 \ge 2$, we can conclude

$$\begin{aligned}(2r+1)(L_j^1 - rL_{j-1}^2) &< L_{j-1}^2 + 2(r+1)p_j - r(2r+1)L_{j-1}^2 \\ &= 2(r+1)p_j - (2r^2+r-1)L_{j-1}^2 \\ &\le 2r(r+1) - 2(2r^2+r-1) = 2(1-r^2) \le 0.\end{aligned}$$

Thus $L_j^1 \le rL_{j-1}^2$.

Case 2. Only one job is processed on M_2 before the arrival of job p_j. Moreover, if $L_{j-1}^1 = 0$, then $L_j^1 = p_j \le r \le rL_{j-1}^2$. Hence, suppose there is at least one job scheduled on M_1 before the arrival of p_j. Then we claim $\overline{LB}_j \ge 2$. In fact, if $L_{j-1}^1 \ge 2$, then $T_j = L_{j-1}^1 + L_{j-1}^2 + p_j \ge 4$, which follows directly that $\overline{LB}_j \ge LB_j \ge \frac{T_j}{2} \ge 2$. Otherwise, we have $1 \le L_{j-1}^1 < 2$. This means that only one job is processed on M_1 before the arrival of the job p_j. Hence $j = 3$ and $\overline{LB}_j = L_j^* \ge \lceil \frac{j}{2} \rceil = 2$.

Since $\overline{LB}_j \ge 2$, we can get $L_{j-1}^2 + p_j \le 2r \le r\overline{LB}_j \le \frac{2r+1}{r+1}\overline{LB}_j = \alpha\overline{LB}_j$, where the last inequality holds because of $r < \phi$. It indicates that p_j must have been scheduled on M_2, which contradicts with the definition of p_j. □

From now on, we focus on the case $\phi \le r < 3$. Note in this case, we have $\alpha \ge \phi$ and $\lambda \ge \phi$. We will prove the following lemma by several observations with respect to the load of L_{j-1}^2 and L_{j-1}^1.

Lemma 3.5 *If $\phi \le r < 3$, then $L_j^1 \le \lambda L_{j-1}^2$.*

Claim 3.6 *If $L_{j-1}^2 \ge 3$, then $L_j^1 \le \lambda L_{j-1}^2$.*

Proof From (3), we get $L_{j-1}^2 + p_j > \alpha\overline{LB}_j \ge \alpha LB_j \ge \frac{\alpha}{2}(L_{j-1}^2 + L_j^1)$, or equivalently, $\alpha L_j^1 < (2-\alpha)L_{j-1}^2 + 2p_j$. Together with $\lambda = \frac{\alpha-1}{2-\alpha}$, $p_j \le r$ and $L_{j-1}^2 \ge 3$, it follows that

$$\begin{aligned}\alpha(L_j^1 - \lambda L_{j-1}^2) < (2-\alpha)L_{j-1}^2 + 2p_j - \alpha\lambda L_{j-1}^2 &= 2p_j - \frac{3\alpha-4}{2-\alpha}L_{j-1}^2 \\ &\le 2r - \frac{9\alpha-12}{2-\alpha}. \qquad (5)\end{aligned}$$

If $\phi \le r < 3\phi - 2$, then $\alpha = \phi$. Thus the above inequality can be written as

$$\alpha(L_j^1 - \lambda L_{j-1}^2) < 2r - \frac{9\phi - 12}{2 - \phi} = 2r - (6\phi - 3) \le 2(3\phi - 2) - (6\phi - 3) = -1 < 0.$$

If $3\phi - 2 \le r < 3$, then $\alpha = \frac{r+2}{3}$ and thus (5) can be continued with

$$\alpha(L_j^1 - \lambda L_{j-1}^2) < 2r - \frac{9 \cdot \frac{r+2}{3} - 12}{2 - \frac{r+2}{3}} = 2r - \frac{9(r-2)}{4-r} = \frac{18 - r - 2r^2}{4-r} < 0,$$

where the last inequality is due to $r \ge 3\phi - 2 > \frac{\sqrt{145}-1}{4}$. In a word, we always have $\alpha(L_j^1 - \lambda L_{j-1}^2) < 0$. Hence the desired result follows accordingly. □

By the above claim, we will always assume in the following discussion that $L_{j-1}^2 < 3$. Let l_i, $i = 1, 2$, be the number of the jobs on M_i before J_j arrives. From Lemma 3.2 and $L_{j-1}^2 < 3$, it is clear that $l_1 \in \{0, 1, 2\}$ and $l_2 \in \{1, 2\}$. According to the values of l_i, we distinguish them into five cases, which will be considered in the following claims.

Claim 3.7 *If $l_1 = 0$, then $L_j^1 \le \lambda L_{j-1}^2$.*

Proof Now we have $L_{j-1}^1 = 0$, $L_j^1 = p_j$. If $L_{j-1}^2 \ge 2$, then by $\phi < 2$ and $3\phi - 2 > 1 + \sqrt{3}$, we get

$$L_j^1 = p_j \le r \le \begin{cases} 3\phi - 2 \le 2\phi \le \phi L_{j-1}^2, & \phi \le r < 3\phi - 2 \\ \frac{2(r-1)}{4-r} \le \frac{r-1}{4-r} L_{j-1}^2, & 3\phi - 2 \le r < 3 \end{cases} = \lambda L_{j-1}^2.$$

If $L_{j-1}^2 < 2$, then only one job is scheduled on M_2 before the arrival of p_j, i.e., $l_2 = 1$. Therefore, $j = 2$ and we get $\overline{LB}_j = L_j^* = \max\{L_{j-1}^2, p_j\} = p_j$ by (4). From (3), we have $L_{j-1}^2 + p_j > \alpha \overline{LB}_j = \alpha p_j$, which, together with $\alpha \ge \phi$, leads to $L_j^1 = p_j < \frac{1}{\alpha - 1} L_{j-1}^2 \le \frac{\alpha - 1}{2 - \alpha} L_{j-1}^2 = \lambda L_{j-1}^2$. □

Claim 3.8 *If $l_1 = l_2 = 1$, then $L_j^1 \le \lambda L_{j-1}^2$.*

Proof Since $l_1 = l_2 = 1$, we have $j = 3$ and jobs p_1 and p_2 are scheduled on different machines. Suppose that the job on M_1 is of hierarchy 1, it is obvious that the algorithm produces the same assignment for the first two jobs no matter which one arrives first. In this case, since p_j (i.e., p_3) is scheduled on M_1, exchanging the arrivals of the job on M_1 and p_3 will not change the schedule and thus we can reduce this case to the case in Claim 3.7. Hence, we only need to consider that both p_1 and p_2 are of hierarchy 2 and clearly p_1 is scheduled on M_2 by the algorithm.

Noting that $j = 3$, we have $\overline{LB}_3 = L_3^*$. Since p_2 is assigned to M_1 by the algorithm, it must be true that $L_2^2 + L_2^1 = p_1 + p_2 > \alpha\overline{LB}_2 = \alpha \max\{L_2^2, L_2^1\} = \alpha L_2^2$, where the last equality holds because of $L_2^1 < L_2^2$ by Lemma 3.2. Consequently, we have

$$L_2^1 > (\alpha - 1)L_2^2. \tag{6}$$

Case 1. If $L_2^2 \le p_3$, then $\overline{LB}_3 = L_3^* = \max\{p_3, L_2^2 + L_2^1\}$. If $L_2^2 + L_2^1 \le p_3$, then $\overline{LB}_3 = p_3$. By (3), we have $L_2^2 > (\alpha-1)p_3$. Thus $p_3 \ge L_2^2+L_2^1 > \alpha(\alpha-1)p_3$, which contradicts with $\alpha(\alpha-1) \ge 1$ from $\alpha \ge \phi$. If $L_2^2 + L_2^1 > p_3$, then $\overline{LB}_3 = L_2^2 + L_2^1$. Again by (3), we have $L_2^2+p_3 > \alpha(L_2^2+L_2^1)$. Thus $L_2^2+L_2^1 > p_3 > \alpha(L_2^2+L_2^1)-L_2^2$. Combining it with (6), we can conclude $L_2^2 > \frac{\alpha-1}{2-\alpha}L_2^1 > \frac{(\alpha-1)^2}{2-\alpha}L_2^2$, which is also a contradiction since $\frac{(\alpha-1)^2}{2-\alpha} \ge 1$ by $\alpha \ge \phi$.

Case 2. If $L_2^2 > p_3$, then $\overline{LB}_3 = L_3^* = \max\{L_2^2, p_3 + L_2^1\}$. A similar argument as Case 1 can be made. If $L_2^2 \ge p_3+L_2^1$, then $\overline{LB}_3 = L_2^2$. By (3), we have $L_2^2+p_3 > \alpha L_2^2$. Together with (6), we get $L_2^2 \ge p_3+L_2^1 > 2(\alpha-1)L_2^2$, which contradicts with $\alpha > \frac{3}{2}$. If $L_2^2 < p_3+L_2^1$, then $\overline{LB}_3 = p_3+L_2^1$. Again by (3), we have $L_2^2+p_3 > \alpha(p_3+L_2^1)$. Thus $p_3+L_2^1 > L_2^2 > (\alpha-1)p_3+\alpha L_2^1$, or equivalently, $p_3 > \frac{\alpha-1}{2-\alpha}L_2^1$. Consequently, $L_2^2 > (\alpha-1)p_3+\alpha L_2^1 > \frac{(\alpha-1)^2}{2-\alpha}L_2^1+\alpha L_2^1 = \frac{1}{2-\alpha}L_2^1$. Combining it with (6), we can conclude $L_2^2 > \frac{\alpha-1}{2-\alpha}L_2^2$, which is also a contradiction since $\alpha > \frac{3}{2}$. □

Claim 3.9 *If $l_1 = 2$ and $l_2 = 1$, then $L_j^1 \le \lambda L_{j-1}^2$.*

Proof In this case, there are two jobs on M_1, denoted by p_i and p_k (p_i arrives before p_k), and one job on M_2, denoted by p_t, before the arrival of p_j. If p_i is the first job, then it must be of hierarchy 1. By exchanging the arrival of p_i and p_t, the schedule produced by the algorithm will not change. Therefore, we can always assume that p_t is the first job and p_i is the second job. If p_i is of hierarchy 1, then exchanging the arrival of p_i and p_k must lead to the same assignment by the algorithm. Similarly, if p_k is of hierarchy 1, then exchanging the arrival of p_k and p_j also leads to the same assignment. In other words, we only need to consider two kinds of instances. The first one is that only one job is scheduled on M_1 before the arrival of p_j, and the second one is that both p_i and p_k are of hierarchy 2. For the first one, we are already done by Claim 3.8. For the second one, we show that the case cannot happen. In fact, if both p_i and p_k are of hierarchy 2, then by the algorithm's rule, $p_i + p_t > \alpha\overline{LB}_2$ and $p_k + p_t > \alpha\overline{LB}_3$. However, on the other hand, by $p_i + p_k < p_t$ from Lemma 3.2, we have $\overline{LB}_2 = L_2^* = p_t$, $\overline{LB}_3 = L_3^* = p_t$. Thus it follows $p_i > (\alpha - 1)p_t$ and $p_k > (\alpha - 1)p_t$, and consequently $p_t > p_i + p_k > 2(\alpha - 1)p_t$, which is a contradiction since $\alpha > \frac{3}{2}$. □

Claim 3.10 *If $l_1 = 1$ and $l_2 = 2$, then $L_j^1 \le \lambda L_{j-1}^2$.*

Proof In this case, there are two jobs on M_2 and one job on M_1 before the arrival of p_j. Let p_i, p_k ($p_i \le p_k$) be those on M_2 and p_t be the one on M_1. Since $j = 4$, we know that $\overline{LB}_j = L_j^*$. Firstly, we prove that $L_{j-1}^2 = p_i + p_k > p_j$. In fact, if $p_i + p_k \le p_j$, then by Lemma 3.2, we have

$$p_t = L^1_{j-1} < L^2_{j-1} = p_i + p_k \le p_j \le r < 3, \tag{7}$$

which indicates that p_j is the current largest job.

Case 1. $L^1_{j-1} = p_t \le p_i$. Therefore, we obtain that $p_t \le p_i \le p_k < p_j$, which, together with (7), leads to $\overline{LB}_j = L^*_j = \max\{p_t + p_i + p_k, p_j\} = p_t + p_i + p_k = L^2_{j-1} + L^1_{j-1}$. By (3), we have $L^2_{j-1} + p_j > \alpha \overline{LB}_j = \alpha(L^2_{j-1} + L^1_{j-1})$. Therefore,

$$\begin{aligned} p_j &> \alpha L^1_{j-1} + (\alpha - 1)L^2_{j-1} \ge \alpha + 2(\alpha - 1) = 3\alpha - 2 \\ &= \begin{cases} 3\phi - 2 > r, & \phi \le r < 3\phi - 2, \\ 3 \cdot \frac{r+2}{3} - 2 = r, & 3\phi - 2 \le r < 3, \end{cases} \end{aligned}$$

which is a contradiction.

Case 2. $L^1_{j-1} = p_t > p_i$. Then we have $p_i \le p_t$, $p_k < p_j$. If $p_t + p_k \le p_j$, then it is clear that $\overline{LB}_j = L^*_j = \max\{p_i + p_t + p_k, p_j\} = p_i + p_t + p_k = L^2_{j-1} + L^1_{j-1}$ by (7). Hence, the same arguments as Case 1 hold, where a contradiction can be found. On the other hand, if $p_t + p_k > p_j$, then clearly $\overline{LB}_j = L^*_j = \max\{p_t + p_k, p_j + p_i\} \ge p_j + p_i$. Combining it with (3), we can conclude that $p_k + (p_j + p_i) = L^2_{j-1} + p_j > \alpha \overline{LB}_j \ge \alpha(p_j + p_i)$, which implies $p_k > (\alpha - 1)(p_j + p_i)$ directly. On the other hand, we have $p_k \le p_j - p_i$ by (7). Thus it implies that $(\alpha - 1)(p_j + p_i) < p_j - p_i$ or equivalently that

$$\frac{p_j}{p_i} > \frac{\alpha}{2 - \alpha} = \begin{cases} \frac{\phi}{2-\phi} > 3\phi - 2 > r, & \phi \le r < 3\phi - 2; \\ \frac{r+2}{4-r} > r, & 3\phi - 2 \le r < 3, \end{cases}$$

which is also a contradiction.

Therefore, we must have $L^2_{j-1} = p_i + p_k > p_j$. Moreover, we claim that $L^1_j = L^1_{j-1} + p_j \le \phi L^2_{j-1}$. Otherwise, we have $T_j = L^1_j + L^2_{j-1} \ge (\phi + 1)L^2_{j-1}$, which follows $\overline{LB}_j \ge LB_j \ge \frac{T_j}{2} \ge \frac{\phi+1}{2} L^2_{j-1}$. Combing it with $\alpha \ge \phi > \frac{3}{2}$ and (3), we can conclude that

$$p_j > \alpha \overline{LB}_j - L^2_{j-1} \ge \frac{\phi(\phi + 1)}{2} L^2_{j-1} - L^2_{j-1} = \left(\phi - \frac{1}{2}\right) L^2_{j-1} > L^2_{j-1},$$

which is a contradiction. Therefore, we must have $L^1_j = L^1_{j-1} + p_j \le \phi L^2_{j-1} \le \lambda L^2_{j-1}$, where the last inequality is due to $\lambda \ge \phi$. Hence we are done. □

Claim 3.11 *The case of $l_1 = l_2 = 2$ cannot happen.*

Proof If it is the case, then there are exactly two jobs scheduled on both machines before the arrival of p_j. Thus, we have $L^1_{j-1} \ge 2$ and $L^2_{j-1} \ge 2$. Since $j = 5$, we can obtain $\overline{LB}_j = L^*_j$. If $\overline{LB}_j \ge 4$, then by (3) and $L^2_{j-1} < 3$, we have $p_j > \alpha \overline{LB}_j - L^2_{j-1} > 4\alpha - 3 > 3$, where the last inequality is due to $\alpha > \frac{3}{2}$. Clearly this is a contradiction. Hence it must be valid that $\overline{LB}_j < 4$.

Since $\overline{LB}_j = L_j^* < 4$, the job p_j has to share its machine with another job in the optimal schedule. It follows that $\overline{LB}_j = L_j^* \geq p_j + 1$, or equivalently,

$$p_j \leq \overline{LB}_j - 1. \tag{8}$$

By $L_{j-1}^2 < 3$ and (3), we have $p_j > \alpha\overline{LB}_j - L_{j-1}^2 > \alpha\overline{LB}_j - 3 \geq \phi\overline{LB}_j - 3$. Combing it with (8), we get $\phi\overline{LB}_j - 3 < p_j \leq \overline{LB}_j - 1$. Hence $\overline{LB}_j < \frac{2}{\phi-1} = 2\phi$. On the other hand, by (3) and $L_{j-1}^1 \geq 2$, we have

$$\overline{LB}_j \geq LB_j \geq \frac{T_j}{2} = \frac{p_j + L_{j-1}^2 + L_{j-1}^1}{2} > \frac{\alpha\overline{LB}_j + 2}{2} \geq \frac{\phi\overline{LB}_j + 2}{2},$$

which indicates that $\overline{LB}_j > \frac{2}{2-\phi} = 2(1+\phi)$. Thus we get $2(1+\phi) < \overline{LB}_j < 2\phi$, also a contradiction! □

From Claims 3.6, 3.7, 3.8, 3.9, 3.10 and 3.11, Lemma 3.5 follows. Finally, by Lemmas 3.3, 3.4, 3.5 and Theorems 3.1, 2.2, the main result of this section can be summarized as follows.

Theorem 3.12 *For any $1 \leq r < 3$, algorithm A has a competitive ratio of α and is optimal.*

4 An optimal preemptive algorithm without introducing idle time

This section considers the preemptive version where idle time is not allowed. We provide an optimal algorithm as follows:

Algorithm B:

1. Let $L_0^1 = L_0^2 = 0$ and $j = 1$.
2. If $p_j \in P_1$, schedule p_j onto machine M_1 at time L_{j-1}^1.
3. Compute the value of L_j^* according to Lemma 2.1.
   - 3.1 If $L_{j-1}^2 + p_j \leq \beta L_j^*$, schedule p_j onto machine M_2 completely at time L_{j-1}^2, go to 4.
   - 3.2 If $L_{j-1}^2 + p_j > \beta L_j^*$, schedule the part $\beta L_j^* - L_{j-1}^2$ of p_j onto machine M_2 at time L_{j-1}^2 and the leftover onto machine M_1 at time L_{j-1}^1, go to 4.
4. Let $j = j + 1$. If no new job arrives, stop. Otherwise, go back to 2.

Clearly, in order to show the feasibility of algorithm B, we only need to prove that the assignment of the job p_j scheduled by step 3.2 is feasible, that is, the time slots assigned to p_j on two machines do not overlap. Furthermore, to obtain that the competitive ratio of B is β, it suffices to verify $\frac{L_j^B}{L_j^*} \leq \beta$ for p_j with hierarchy 1, since the assignment of p_j with hierarchy 2 satisfies $\frac{L_j^B}{L_j^*} \leq \beta$ obviously (because of the algorithm rule of step 3). The detailed arguments begin with the following lemma.

Lemma 4.1 *Algorithm B is feasible.*

Proof As stated above, we only need to show that the time slots assigned to p_j in step 3.2 do not overlap, which is equivalent to show

$$L_j^1 = L_{j-1}^1 + p_j - (\beta L_j^* - L_{j-1}^2) \leq L_{j-1}^2,$$

i.e.,

$$L_{j-1}^1 + p_j - \beta L_j^* \leq 0. \tag{9}$$

Clearly, if $j = 2$, by the algorithm rule, we can obtain that $L_1^1 = 0$(when $g_1 = 2$) or alternatively $L_1^2 = 0$(when $g_1 = 1$). For the former, (9) holds clearly because of $L_2^* = \max\{p_1, p_2\}$ and $\beta > 1$. For the latter, p_2 should be scheduled by step 3.1, a contradiction.

Now we turn to consider the case $j \geq 3$. We prove (9) by contradiction. Suppose

$$L_{j-1}^1 + p_j > \beta L_j^*. \tag{10}$$

Note that the algorithm rule of step 3.2 implies

$$L_{j-1}^2 + p_j > \beta L_j^*. \tag{11}$$

By (10) and (11) and Lemma 2.1, we have

$$L_{j-1}^1 + p_j + L_{j-1}^2 + p_j > 2\beta L_j^* \geq 2\beta \frac{T_{j-1} + p_j}{2} = \beta(T_{j-1} + p_j),$$

which, together with the fact that $T_{j-1} \geq 2$ due to $j \geq 3$, implies that

$$p_j > \frac{\beta - 1}{2 - \beta} T_{j-1} = \frac{r}{2} T_{j-1} \geq r,$$

a contradiction. Hence (9) holds. □

Before going to achieve the desired competitive ratio, we first give the following lemma.

Lemma 4.2 *For any j, if there exist jobs with hierarchy 2 processed on machine M_1, and let p_l, $l \leq j$, be the last one (or a part of which) in those jobs, then we have*

$$\frac{L_l^1}{L_l^2} \leq \frac{r}{2}.$$

Proof From the algorithm description, we can see that the job p_l must be processed by step 3.2. It follows that $L_l^2 = \beta L_j^*$ and $L_l^1 = T_l - L_l^2$. Note that $L_l^* \geq \frac{T_l}{2}$, we have

$$\frac{L_l^1}{L_l^2} = \frac{T_l - L_l^2}{L_l^2} = \frac{T_l}{\beta L_j^*} - 1 \leq \frac{2}{\beta} - 1 = \frac{1}{r+1} \leq \frac{r}{2},$$

where the last inequality holds because of $r \geq 1$. □

Theorem 4.3 *The competitive ratio of algorithm B is β. Thus it is optimal.*

Proof If the makespan of algorithm B is determined by a job p_j with $g_j = 2$, it is not hard to obtain that $C^B = L_j^2$ by the assignment of the job p_j. From the algorithm description in steps 3.1 and 3.2 we have $\frac{C^B}{C^*} \leq \beta$. On the other hand, suppose the makespan of algorithm B is determined by a job p_j with hierarchy of 1. Then $C^B = L_j^1$.

If there does not exist any job with hierarchy 2 to be processed on machine M_1, then we can conclude that $C^B = T_{j1} \leq C^*$ by Lemma 2.1. Therefore, we assume that there exist some jobs with hierarchy 2 on machine M_1. We denote $T = L_j^1 - L_l^1 = C^B - L_l^1$, where l is defined in the same way as Lemma 4.2.

By Lemma 2.1, we have $C^* \geq \max\{\frac{L_l^1 + L_l^2 + T}{2}, T\}$. Note that $\frac{L_l^1}{L_l^2} \leq \frac{r}{2}$ by Lemma 4.2. Therefore,

$$\begin{aligned}
\frac{C^B}{C^*} &\leq \frac{L_l^1 + T}{C^*} \\
&= \begin{cases} \frac{L_l^1 + T}{T}, & \text{if} \quad L_l^1 + L_l^2 \leq T, \\ \frac{L_l^1 + T}{\frac{L_l^1 + L_l^2 + T}{2}}, & \text{if} \quad L_l^1 + L_l^2 > T, \end{cases} \\
&\leq 1 + \frac{L_l^1}{L_l^1 + L_l^2} \\
&\leq 1 + \frac{1}{1 + 2/r} = \beta.
\end{aligned}$$

By now we have completed the proof of the competitive ratio. Moreover, the optimality of algorithm B is a direct consequence of Theorem 2.4. □

5 Conclusions

We studied the online hierarchical scheduling problem on two parallel identical machines with the sizes of all jobs in $[p, rp]$, where $p > 0$ and $r \geq 1$. For non-preemptive version, we proposed an optimal algorithm for all possible r. For preemptive version, we showed that the semi-online information is useless if idle time is allowed. Moreover, optimal preemptive algorithm that does not use idle time is also presented.

We feel that it is interesting to extend the result to $m(m > 2)$ identical machines, as well as two uniform machines. In addition, it is also interesting to study what kind of information it needs to improve preemptive algorithms.

Acknowledgments An Zhang, Yiwei Jiang, and Jueliang Hu were supported by the National Natural Science Foundation of China through Grant Nos. 11201105, 11001242, and 11071220 respectively. Yiwei Jiang is also supported by Zhejiang Province Natural Science Foundation of China (Y6090175).

References

Bar-Noy A, Freund A, Naor J (2001) On-line load balancing in a hierarchical server topology. SIAM Journal on Computing. 31:527–549

Chassid O, Epstein L (2008) The hierarchical model for load balancing on two machines. Journal of Combinatorial Optimization. 15(4):305–314

Crescenzi P, Gambosi G, Penna P (2004) On-line algorithms for the channel assignment problem in cellular networks. Discrete Applied Mathematics. 137:237–266

Dósa G, Epstein L (2008) Preemptive scheduling on a small number of hierarchical machines. Information and Computation. 206(5):602–619

He Y, Zhang G (1999) Semi on-line scheduling on two identical machines. Computing. 62:179–187

Jiang Y (2008) Online scheduling on parallel machines with two GoS levels. Journal of Combinatorial Optimization. 16:28–38

Jiang Y, He Y, Tang C (2006) Optimal online algorithms for scheduling on two identical machines under a grade of service. Journal of Zhejiang University Science. 7A:309–314

Liu M, Chu C, Xu Y, Zheng F (2011) Semi-online scheduling on 2 machines under a grade of service provision with bounded processing times. Journal of Combinatorial Optimization. 21:138–149

Park J, Chang S, Lee K (2006) Online and semi-online scheduling of two machines under a grade of service provision. Operations Research Letters. 34:692–696

Tan Z, Zhang A (2010) A note on hierarchical scheduling on two uniform machines. Journal of Combinatorial Optimization. 20:85–95

Tan Z, Zhang A (2011) Online hierarchical scheduling: An approach using mathematical programming. Theoretical Computer Science. 412:246–256

Wu Y, Ji M, Yang Q (2012) Optimal semi-online scheduling algorithms on two parallel identical machines under a grade of service provision. International Journal of Production Economics. 135:367–371

Zhang A, Jiang Y, Tan Z (2009) Online parallel machines scheduling with two hierarchies. Theoretical Computer Science. 410:3597–3605