Theoretical Computer Science 531 (2014) 37–46

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Optimal algorithms for semi-online machine covering on two hierarchical machines

Yong Wu [a,b], T.C.E. Cheng [b], Min Ji [c,*]

[a] Department of Fundamental Education, Ningbo Institute of Technology, Zhejiang University, Ningbo 315100, PR China
[b] Department of Logistics and Maritime Studies, The Hong Kong Polytechnic University, Kowloon, Hong Kong
[c] School of Computer Science and Information Engineering, Contemporary Business and Trade Research Center, Zhejiang Gongshang University, Hangzhou 310018, PR China

ARTICLE INFO

Article history:
Received 16 October 2013
Accepted 14 February 2014
Communicated by D.-Z. Du

Keywords:
Scheduling
Semi-online
Hierarchy
Two machines
Competitive ratio

ABSTRACT

This paper investigates the semi-online machine covering problem on two hierarchical machines where the jobs are correspondingly classified into two hierarchical classes. The objective is to maximize the minimum machine load. We show that if we only know the size of the largest job, no algorithm with a bounded competitive ratio exists. So we consider the case where we know both the size and the class of the largest job. If we know the size of the largest job and that it belongs to the higher class, then an optimal algorithm with a $(1+\frac{\sqrt{2}}{2})$-competitive ratio exists. If we know the size of the largest job and that it belongs to the lower class, we design an optimal algorithm with an α-competitive ratio, where $\alpha \approx 2.48119$ is the largest root of the equation $x^3-2x^2-2x+2=0$. For the case where the total size of all the jobs is known in advance, we show that the competitive ratio of an optimal algorithm is 2.

1. Introduction

In this paper we study semi-online variants of the machine covering problem on two hierarchical machines where the jobs are correspondingly classified into two hierarchical classes. The goal is to maximize the minimum machine load under the constraint that all the requests are satisfied. First proposed by Deuermeyer et al. [5], the machine covering problem has application in the sequencing of maintenance activities for modular gas turbine aircraft engines. Bar-Noy et al. [1] first studied hierarchical scheduling. It is a common practice in the service industry that differentiated services are provided to customers based on their entitled privileges that are assigned according to their classes in the service hierarchy. While hierarchy is a subjective concept, it is often put into practice in terms of different levels of access privilege to service capacity. Hierarchical scheduling has many applications, such as in the service industry, computer systems, hierarchical databases etc.

We focus on semi-online algorithms in this paper. In online and semi-online scheduling, the performance of an algorithm is often measured by its *competitive ratio*. For a problem instance $\mathcal{J}$ and an algorithm A, let $C^A(\mathcal{J})$ (or C^A in short) be the objective value produced by A and let $C^*(\mathcal{J})$ (or C^* in short) be the optimal value of the corresponding offline version (i.e., the optimal offline value). Then the competitive ratio of A is the smallest number c such that for any instance $\mathcal{J}$, $C^*(\mathcal{J}) \leqslant cC^A(\mathcal{J})$. If the competitive ratio of an algorithm is at most α, we say that the algorithm is α-*competitive*.

* Corresponding author.
E-mail addresses: wuyong@nit.zju.edu.cn (Y. Wu), edwin.cheng@polyu.edu.hk (T.C.E. Cheng), jimkeen@163.com (M. Ji).

http://dx.doi.org/10.1016/j.tcs.2014.02.015
0304-3975/

If no c satisfying the inequality exists, we say that the competitive ratio is unbounded or ∞. An online (semi-online) scheduling problem has a *lower bound* ρ if no online (semi-online) algorithm has a competitive ratio smaller than ρ. An online (semi-online) algorithm A is called *optimal* if its competitive ratio matches the lower bound for the problem.

In recent years, there have been many results on the study of hierarchical scheduling. Hwang et al. [7] study offline hierarchical scheduling to minimize the makespan and propose an approximation algorithm *LG-LPT*. They prove that its makespan is not greater than 5/4 times the optimal makespan for $m = 2$ and not greater than $2 - \frac{1}{m-1}$ times the optimal makespan for $m \geqslant 3$, where m is the number of hierarchical machines. Glass and Kellerer [6] give an improved algorithm with a worst-case ratio at most 3/2 for m machines. Ji and Cheng [8] propose a fully polynomial-time approximation scheme (FPTAS) for hierarchical scheduling to minimize the makespan on m parallel machines.

For online hierarchical scheduling to minimize the makespan, Bar-Noy et al. [1] first present an $(e + 1)$-competitive algorithm for the general case with m machines, which Crescenzi et al. [4] also provide. For the case with two machines, Park et al. [12] and Jiang et al. [10] independently propose an optimal algorithm with a competitive ratio 5/3. Jiang [9] extends the result to the case where there are exactly two hierarchical job classes on m machines. He proves that 2 is a lower bound for online algorithms and proposes an online algorithm with a competitive ratio $(12 + 4\sqrt{2})/7$. Zhang et al. [16] improve the ratio to $1 + \frac{m^2-m}{m^2-mk+k^2} \leqslant 7/3$, where k is the number of machines that can process the high class jobs.

First to study semi-online hierarchical scheduling to minimize the makespan, Park et al. [12] propose an optimal algorithm with a competitive ratio 3/2 for the case where the total size of all the jobs is known in advance. Liu et al. [11] study the case with bounded jobs, i.e., the processing time of each job is bounded within an interval $[a, \alpha a]$. Recently, Zhang et al. [15] provide optimal algorithm for the problem. Wu et al. [14] consider the cases where the optimal offline value of the instance is known in advance and where the largest size of the jobs is known in advance. They provide optimal algorithms for both problems. Extending hierarchical scheduling to the case with two uniform machines, Chassid and Epstein [2] study online and semi-online problems and provide optimal algorithms.

In this paper we consider the semi-online hierarchical machine covering problems on two parallel identical machines. We study two cases where the total size of all the jobs (denoted by T) is known in advance and where the size of the largest job (denoted by p_{max}) is known in advance. T and p_{max} are often assumed to be known in advance in the semi-online scheduling literature for various reasons as stated in [3], and [13]. For the case where p_{max} is known in advance, we show that there exists no algorithm with a bounded competitive ratio. In order to overcome this barrier, we assume that both the size and class of the largest job are known in advance, i.e., we know $J_{max} = (p_{max}, g_{max})$ in advance. For the case where $g_{max} = 1$, we design an optimal algorithm with a competitive ratio $1 + \frac{\sqrt{2}}{2}$. For the case where $g_{max} = 2$, we design an optimal algorithm with a competitive ratio α, where $\alpha \approx 2.48119$ is the largest root of the equation $x^3 - 2x^2 - 2x + 2 = 0$. Finally, we study the case where T is known in advance and provide an optimal algorithm with a competitive ratio 2 for the problem.

The rest of the paper is organized as follows: In Section 2 we introduce the notation and formulate the problems. In Section 3 we propose optimal algorithms for the case where we know the largest job and its class in advance. In Section 4 we provide an optimal algorithm for the case where we know the total size of the jobs in advance. Finally, we conclude the paper and suggest topics for future research in Section 5.

2. Problem definitions

We are given two parallel identical machines M_1 and M_2, and a set $\mathcal{J}$ of n independent jobs $J_1, J_2, \ldots, J_n$. We denote each job by $J_i = (p_i, g_i)$, where p_i is the size of J_i and $g_i \in \{1, 2\}$ is the class of J_i. Machine M_k has a certificate $g(M_k) = k$, $k = 1, 2$, associated with it. Machine M_k can process J_i only when $g(M_k) \leqslant g_i$. p_i and g_i are not known until the arrival of job J_i. Each job J_i emerges immediately after J_{i-1} is scheduled. Let $G_1 = \{J_i \mid g_i = 1\}$ and $G_2 = \{J_i \mid g_i = 2\}$, so $\mathcal{J} = G_1 \cup G_2$. We define the *load* of a machine as the completion time of the machine, i.e., the total size of all the jobs processed on it. Let L_1 and L_2 denote the loads of machines M_1 and M_2, respectively. We must assign all the jobs to one of the two machines and the objective value of a schedule is $\min\{L_1, L_2\}$. We state the online problem as follows:

$$\text{Given } \mathcal{J}, \text{ find a schedule to maximize } \min\{L_1, L_2\}.$$

Knowing T (or p_{max}, or J_{max}) in advance, we state the semi-online variant of the problem as follows:

$$\text{Given } \mathcal{J} \text{ and } T \text{ (or } p_{max}, \text{ or } J_{max}), \text{ find a schedule to maximize } \min\{L_1, L_2\}.$$

To ease presentation and exposition, we introduce the following notation for use in the remainder of the paper.

- G_i^k is the set of jobs of class i, $i = 1, 2$, in the first k jobs.
- G_{2i}^k is the set of jobs of class 2 assigned to machine M_i, $i = 1, 2$, immediately after job J_k is assigned.
- $t(\delta)$ is the total size of the jobs in any job set δ.
- $t(G_i^k)$ is the total size of the jobs in the job set G_i^k, $i = 1, 2$. It follows that $t(G_i^n) = t(G_i)$, $i = 1, 2$.
- p_{max} is the largest size of all the jobs.
- $J_B = (p_{max}, g_{max})$ is the first largest job with size p_{max} and of class g_{max} that we know in advance.

3. Largest job is known

We first show in Lemma 1 that if we only know the size of the largest job, then no algorithm with a bounded competitive ratio exists. In addition, the job instances used in the proof of Lemma 1 also prove that no algorithm with a bounded competitive ratio exists for the online machine covering problem on two hierarchical machines. Thus more specific models need to be studied.

Lemma 1. *Any semi-online algorithm for the problem, where we know p_{max} in advance, has an unbounded competitive ratio.*

Proof. Consider an algorithm A and the following sequence of jobs. Let ϵ be a sufficiently small number. The first job is $J_1 = (\epsilon, 2)$ and we must assign it to machine M_2, else a second job $J_2 = (p_{max}, 1)$ arrives. Job J_2 must be assigned to machine M_1, and we get $C^A = 0$ and $C^* = \epsilon$. It follows that $C^*/C^A = \infty$.

Otherwise, the second job is $J_2 = (p_{max}, 2)$ and we must assign it to machine M_1, else we get $C^A = 0$, so $C^* = \epsilon$ and $C^*/C^A = \infty$ again.

Finally, a third job $J_3 = (p_{max}, 1)$ arrives. We must assign it to machine M_1 together with job J_2. At this point, $C^A = \epsilon$ while $C^* = p_{max}$, which yields $C^*/C^A \to \infty$ as $\epsilon \to 0$. □

The following lemma generalizes an upper bound for the off-line optimal value C^*. This result is useful throughout the paper.

Lemma 2. *For the machine covering problem on two hierarchical machines, we have $C^* \leqslant \min\{t(G_2), (t(G_1) + t(G_2))/2\}$.*

Proof. If $t(G_2) \leqslant t(G_1)$, then $C^* \leqslant t(G_2)$ since all the jobs of class 1 in G_1 must be assigned to machine M_1. If $t(G_2) > t(G_1)$, then $C^* \leqslant T/2 \leqslant (t(G_1) + t(G_2))/2$ and we get the result. □

In the next two subsections, we study the case where we know the largest job and its class, i.e., $J_{max} = \{p_{max}, g_{max}\}$ in advance. We design an optimal algorithm for each case depending on the class of the largest job.

3.1. Largest job of class 1

Theorem 1. *A lower bound for the case where we know $J_{max} = (p_{max}, 1)$ in advance is at least $1 + \frac{\sqrt{2}}{2}$.*

Proof. First, we declare that the size of the largest job is 1, i.e., $p_{max} = 1$. We begin with jobs $J_1 = (1, 1)$ and $J_2 = (x, 2)$, where the exact value of $x (\leqslant 1/2)$ will be decided later. Job J_1 must be assigned to machine M_1. If job J_2 is assigned to machine M_1, then no more jobs arrive. At this point, $C^* = x$ and we have $C^A = 0$, so $C^*/C^A \to \infty$. Thus job J_2 must be assigned to machine M_2 and we generate job $J_3 = (x, 2)$. If job J_3 is scheduled on machine M_1, then no more jobs arrive, which yields $C^*/C^A \geqslant 2$. Otherwise, if job J_3 is scheduled on machine M_2, then we generate job $J_4 = (2x, 2)$. If job J_4 is assigned to M_1, then job $J_5 = (1, 1)$ arrives and we have $C^*/C^A \geqslant 2$. Otherwise, job J_4 is assigned to M_2 and we generate the remaining job(s) with $J_5 = (1, 2)$ (when J_5 is assigned to M_2) or $J_5 = (1, 2)$, $J_6 = (1, 1)$, and $J_7 = (1, 1)$ (when J_5 is assigned to M_1). Thus, we have $C^*/C^A \geqslant \min\{1 + 2x, 1 + 1/4x\}$. Letting $x = \sqrt{2}/4$, we have $C^*/C^A \geqslant 1 + \sqrt{2}/2$. □

Letting $\alpha = 1 + \sqrt{2}/2$, we present the following semi-online algorithm and prove that it is optimal with a competitive ratio α.

Algorithm *HM*1.

1. Let $J_i = \{p_i, g_i\}$ be the current job;
2. If $g_i = 1$, then assign J_i to machine M_1;
3. If $g_i = 2$ and $t(G_{21}^{i-1}) + p_i \leqslant (1 - 1/\alpha) t(G_2^i)$, then assign J_i to machine M_1; otherwise, assign J_i to machine M_2;
4. If no more jobs arrive, then stop; else, let $i = i + 1$ and go to Step 2.

Theorem 2. *The competitive ratio of Algorithm HM1 for the problem is at most $\alpha = 1 + \sqrt{2}/2$.*

Proof. According to Algorithm *HM*1, we have $t(G_{21}^n) \leqslant (1 - 1/\alpha) t(G_2^n)$. It follows that $L_2 = t(G_{22}^n) = t(G_2) - t(G_{21}^n) \geqslant t(G_2)/\alpha$. Lemma 2 shows that $C^* \leqslant t(G_2)$, so $C^*/L_2 \leqslant \alpha$. If we also have $C^*/L_1 \leqslant \alpha$, then we get the result. Thus, we assume that $C^*/L_1 > \alpha$, i.e., $L_1 = t(G_{21}^n) + t(G_1^n) < C^*/\alpha$.

At this point, we have $L_1 < \frac{C^*}{\alpha} \leqslant \frac{L_1 + L_2}{2\alpha}$ by Lemma 2. It follows that

$$L_1 = t(G_1) + t\big(G_{21}^n\big) < \frac{1}{2\alpha - 1} L_2. \tag{1}$$

Let job $J_k = \{p_k, 2\}$ be the last one assigned to machine M_2 by Algorithm $HM1$. By inequality (1), we get

$$t(G_1) + t\big(G_{21}^{k-1}\big) \leqslant t(G_1) + t\big(G_{21}^{n}\big) < \frac{1}{2\alpha - 1} L_2 = \frac{1}{2\alpha - 1} t\big(G_{22}^{k}\big) \leqslant \frac{1}{2\alpha - 1} t\big(G_{2}^{k}\big).$$

Noting that there exists a largest job $J_{max} = \{p_{max}, 1\}$ in G_1 and $p_k \leqslant p_{max}$, we have the following inequality

$$p_k + t\big(G_{21}^{k-1}\big) \leqslant t(G_1) + t\big(G_{21}^{k-1}\big) \leqslant \frac{1}{2\alpha - 1} t\big(G_{2}^{k}\big). \tag{2}$$

Inequality $\frac{1}{2\alpha-1} = 1 - \frac{1}{\alpha}$ holds since $\alpha = 1 + \sqrt{2}/2$. By (2), we know that job J_k must be assigned to machine M_1 by Step 3 of Algorithm $HM1$. This contradicts the definition of job J_k. Therefore, we also have $C^*/L_1 \leqslant \alpha$, yielding the result. □

From Theorems 1 and 2, we know that $HM1$ is an optimal algorithm with a competitive ratio α for the case where we know $J_{max} = (p_{max}, 1)$ in advance.

3.2. Largest job of class 2

Theorem 3. *A lower bound for the case where we know $J_{max} = (p_{max}, 2)$ in advance is at least α, where $\alpha \approx 2.48119$ is the largest root of the equation $x^3 - 2x^2 - 2x + 2 = 0$.*

Proof. First, we assume without loss of generality that the size of the largest job is 1, i.e., $p_{max} = 1$. We begin with jobs $J_1 = (1, 2)$ and $J_2 = (\frac{1}{\alpha^2-1}, 2)$. If we assign both jobs to the same machine, then no more jobs arrive and we get $C^*/C^A \to \infty$. If we assign job J_1 to machine M_1, then jobs $J_3 = (1, 1)$ and $J_4 = (\frac{1}{\alpha^2-1}, 1)$ arrive. At this point, we have $C^* = \frac{\alpha^2}{\alpha^2-1}$ and $C^A = \frac{1}{\alpha^2-1}$, so $C^*/C^A = \alpha^2 > \alpha$. Thus we must assign job J_1 to machine M_2 and job J_2 to machine M_1, and we generate job $J_3 = (\frac{\alpha-1}{\alpha^2-1}, 2)$. If job J_3 is scheduled on machine M_2, then no more jobs arrive. It follows that $C^*/C^A \geqslant \alpha$. Otherwise, if job J_3 is scheduled on machine M_1, then we generate job $J_4 = (1, 2)$. If job J_4 is assigned to M_2, then no more jobs arrive, and we have $C^* = 1 + \frac{1}{\alpha^2-1}$ and $C^A = \frac{\alpha}{\alpha^2-1}$, which shows that $C^*/C^A \geqslant \alpha$. Otherwise, job J_4 is assigned to M_1 and we generate jobs $J_5 = (1, 1)$, $J_6 = (1, 1)$, and $J_7 = (\frac{\alpha}{\alpha^2-1}, 1)$. Thus, we have $C^* = 2 + \frac{\alpha}{\alpha^2-1}$ and $C^A = 1$. So $C^*/C^A \geqslant \min\{\alpha, 2 + \frac{\alpha}{\alpha^2-1}\} = \alpha$. □

Let α be the largest root of the equation $x^3 - 2x^2 - 2x + 2 = 0$. We propose Algorithm $HM2$ for this problem. We assume that J_B is the first job of the sequence and always assign it to machine M_2. Such an assumption does not affect the correctness of the result because partial information ensures the existence of J_B. One can easily modify Step 4 of Algorithm $HM2$ given below to solve instances without the above assumption by always taking p_B into account when considering the current sizes of $t(G_{22}^{i-1})$ and $t(G_2^i)$, i.e., modifying the definitions of the current sizes of $t(G_{22}^{i-1})$ and $t(G_2^i)$ right before the assignment of J_i as $\bar{t}(G_{22}^{i-1})$ and $\bar{t}(G_2^i)$, respectively, as follows:

$$\bar{t}\big(G_{22}^{i-1}\big) = \begin{cases} t(G_{22}^{i-1}) + p_{max} & \text{if } J_i \text{ comes before } J_B, \\ t(G_{22}^{i-1}) & \text{otherwise.} \end{cases}$$

$$\bar{t}\big(G_{2}^{i}\big) = \begin{cases} t(G_{2}^{i}) + p_{max} & \text{if } J_i \text{ comes before } J_B, \\ t(G_{2}^{i}) & \text{otherwise.} \end{cases}$$

Algorithm *HM2*.

1. Assign job $J_B = \{p_{max}, 2\}$ to M_2;
2. Let $J_i = \{p_i, g_i\}$ be the current job;
3. If $g_i = 1$, then assign J_i to machine M_1;
4. If $g_i = 2$, then assign J_i by the following rule:
   4.1 if $t(G_{21}^{i-1}) + p_i \leqslant (\alpha - 2)p_{max}$, then assign J_i to machine M_1;
   4.2 else if $t(G_{22}^{i-1}) + p_i - p_{max} \leqslant (\alpha - 1)t(G_{21}^{i-1})$, then assign J_i to machine M_2;
   4.3 else if $t(G_{21}^{i-1}) + p_i \leqslant (1 - 1/\alpha)t(G_2^i)$, then assign J_i to machine M_1;
   4.4 otherwise, assign J_i to machine M_2;
5. If no more jobs arrive, then stop; else, let $i = i + 1$ and go to Step 2.

Theorem 4. *The competitive ratio of Algorithm HM2 for the problem is at most $\alpha \approx 2.48119$.*

Proof. According to Algorithm *HM*2, we have $t(G_{21}^n) \leqslant (1-1/\alpha)t(G_2^n)$, which yields

$$t\bigl(G_{22}^n\bigr) \geqslant t\bigl(G_2^n\bigr)/\alpha. \tag{3}$$

Next, we distinguish two cases according to the performance of the algorithm.

Case 1. There is no job $J_i = (p_i, 2)$ that makes $t(G_{21}^{i-1}) + p_i > (1-1/\alpha)t(G_2^i)$, $1 \leqslant i \leqslant n$.

In this case, we have only assigned jobs to machine M_2 by Step 4.2 of Algorithm *HM*2. If $C^{HM2} = \min\{L_1, L_2\} = \min\{t(G_1^n) + t(G_{21}^n), t(G_{22}^n)\} = t(G_{22}^n)$, then we get $C^*/C^{HM2} \leqslant \alpha$ by Lemma 2 and inequality (3).

If $C^{HM2} = t(G_1^n) + t(G_{21}^n) < t(G_{22}^n)$, then we have $t(G_2) > t(G_1)$ and $t(G_{22}^n) - p_{max} \leqslant (\alpha-1)t(G_{21}^n)$. Otherwise, if $t(G_{22}^n) - p_{max} > (\alpha-1)t(G_{21}^n)$, then there exists a job assigned to machine M_2 according to Step 4.4 of Algorithm *HM*2. This is a contradiction. In the offline optimal schedule, we must assign job $J_B = \{p_{max}, 2\}$ to machine M_2 and some of the other jobs of class 2 to machine M_1. Thus,

$$C^* \leqslant t(G_2) - p_{max} + t(G_1) = t\bigl(G_{22}^n\bigr) - p_{max} + t\bigl(G_{21}^n\bigr) + t(G_1)$$

and it follows that

$$\frac{C^*}{C^{HM2}} \leqslant \frac{t(G_{22}^n) - p_{max} + t(G_{21}^n) + t(G_1)}{t(G_1^n) + t(G_{21}^n)} \leqslant \frac{\alpha t(G_{21}^n) + t(G_1)}{t(G_1^n) + t(G_{21}^n)} \leqslant \alpha.$$

Case 2. There exists at least one job $J_i = (p_i, 2)$ that satisfies $t(G_{21}^{i-1}) + p_i > (1-1/\alpha)t(G_2^i)$, $1 \leqslant i \leqslant n$.

Let job $J_k = \{p_k, 2\}$ be the first such job, which is assigned to machine M_2 according to Step 4.4 of Algorithm *HM*2. We can derive a lemma about the size of job J_k.

Lemma 3. *Let $t(\beta)$ be the total size of all the jobs assigned to machine M_2 by Step 4.2 of Algorithm HM2 before job J_k, then we have $p_k > (\alpha-1)(p_{max} + t(\beta)) - t(G_{21}^{k-1})$.*

Proof. At this point, $t(G_2^k) = t(G_{21}^{k-1}) + t(G_{22}^{k-1}) + p_k$ and $t(G_{22}^{k-1}) = p_{max} + t(\beta)$. According to the definition of job J_k, we have

$$t\bigl(G_{21}^{k-1}\bigr) + p_k > (1-1/\alpha)t\bigl(G_2^k\bigr) = (1-1/\alpha)\bigl(t\bigl(G_{21}^{k-1}\bigr) + p_{max} + t(\beta) + p_k\bigr).$$

It follows that $p_k > (\alpha-1)(p_{max} + t(\beta)) - t(G_{21}^{k-1})$. $\square$

According to inequality (3) and Lemma 2, we have $L_2 = t(G_{22}^n) \geqslant C^*/\alpha$. Note that $C^{HM2} = \min\{L_1, L_2\}$. If we also have $L_1 = t(G_1) + t(G_{21}^n) \geqslant C^*/\alpha$, then we are done.

Suppose that the theorem is false in this case, i.e.,

$$C^{HM2} = \min\{L_1, L_2\} = t(G_1) + t\bigl(G_{21}^n\bigr) < C^*/\alpha. \tag{4}$$

There must exist a problem instance that we call a counter example for which Algorithm *HM*2 yields a schedule with $C^{HM2} < C^*/\alpha$. Then, the counter example with the least number of jobs is defined as the minimum counter example. For notational ease in the remainder of this paper, we let $\mathcal{J} = \{J_1, J_2, \dots, J_n\}$ be the minimum counter example. We provide a lemma about the minimum counter example.

Lemma 4. *For a minimum counter example $\mathcal{J} = \{J_1, J_2, \dots, J_n\}$, $(\alpha-2)p_{max} < t(G_{21}^n) < \frac{1}{2\alpha-3}p_{max}$ and $t(G_2) < \frac{2\alpha}{2\alpha-3}p_{max}$.*

Proof. According to Lemma 3, we have $p_k + t(G_{21}^n) \geqslant p_k + t(G_{21}^{k-1}) > (\alpha-1)p_{max}$. Note that $p_k \leqslant p_{max}$. So we get $t(G_{21}^n) > (\alpha-2)p_{max}$.

According to Lemma 2 and inequality (4), we have

$$L_1 = t(G_1) + t\bigl(G_{21}^n\bigr) < \frac{1}{2\alpha}\bigl(t(G_1) + t(G_2)\bigr).$$

Note that $t(G_2) = t(G_{21}^n) + t(G_{22}^n)$. So we have

$$t\bigl(G_{22}^n\bigr) > (2\alpha-1)\bigl(t(G_1) + t\bigl(G_{21}^n\bigr)\bigr) \tag{5}$$

and

$$t(G_2) = t\left(G_{22}^n\right) + t\left(G_{21}^n\right) > 2\alpha t\left(G_{21}^n\right). \tag{6}$$

According to Algorithm $HM2$, we have

$$t\left(G_{21}^n\right) > (1 - 1/\alpha)t(G_2) - p_{max}. \tag{7}$$

Otherwise, if $t(G_{21}^n) \leqslant (1 - 1/\alpha)t(G_2) - p_{max}$, then $t(G_{21}^n) + p_{max} \leqslant (1 - 1/\alpha)t(G_2)$. Note that in the minimum counter example, there are at least two jobs assigned to machine M_2. Thus the last job assigned to machine M_2 must be assigned to machine M_1 by Step 4.3.

Together with inequalities (6) and (7), we have $t(G_{21}^n) > (1 - \frac{1}{\alpha}) \cdot 2\alpha t(G_{21}^n) - p_{max}$. It follows that

$$t\left(G_{21}^n\right) < \frac{1}{2\alpha - 3} p_{max}. \tag{8}$$

Next, we prove that $t(G_2) < \frac{2\alpha}{2\alpha-3} p_{max}$. Otherwise, assuming that $t(G_2) \geqslant \frac{2\alpha}{2\alpha-3} p_{max}$, we have $t(G_{21}^n) + p_k > (1 - \frac{1}{\alpha})t(G_2) \geqslant \frac{2\alpha-2}{2\alpha-3} p_{max}$ by the definition of job J_k. Note that $p_k \leqslant p_{max}$. So we have $t(G_{21}^n) \geqslant \frac{1}{2\alpha-3} p_{max}$. This contradicts inequality (8). □

Lemma 5. *For a minimum counter example $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$, there exists only one job J_k that is assigned to machine M_2 by Step 4.4 and job J_k is also the last job assigned to machine M_2.*

Proof. We first show the uniqueness of job J_k. Assume that there is another job $J_a = \{p_a, 2\}$ that is assigned to machine M_2 by Step 4.4. Note that job J_k is the first one, so job J_a arrives after job J_k. Lemmas 3 and 4 show that

$$p_k > (\alpha - 1)p_{max} - t\left(G_{21}^n\right) > \left(\alpha - 1 - \frac{1}{2\alpha - 3}\right)p_{max} \approx 0.971611 p_{max}. \tag{9}$$

According to the definition of job J_a, we have

$$t\left(G_{21}^{a-1}\right) + p_a > \left(1 - \frac{1}{\alpha}\right)t\left(G_2^a\right) = \left(1 - \frac{1}{\alpha}\right)\left(t\left(G_{21}^a\right) + t\left(G_{22}^a\right)\right)$$

and

$$t\left(G_{22}^a\right) \geqslant t\left(G_{22}^k\right) + p_a = \left(p_{max} + t(\beta) + p_k + p_a\right),$$

which yields $t(G_{21}^{a-1}) + p_a > (\alpha - 1)(p_{max} + p_k)$. Note that $t(G_{21}^{a-1}) \leqslant t(G_{21}^n)$. Combining with inequality (9), we have

$$p_a > \alpha\left(\alpha - 1 - \frac{1}{2\alpha - 3}\right)p_{max} \approx 2.41076 p_{max},$$

a contradiction.

Next we show that job J_k is the last one assigned to machine M_2 by Algorithm $HM2$. If this is not true, let job $J_a = \{p_a, 2\}$ be the last job assigned to machine M_2. It is clear that job J_a is assigned to machine M_2 by Step 4.2 of Algorithm $HM2$.

At this point, we have $t(G_{22}^{a-1}) - p_{max} + p_a \leqslant (\alpha - 1)t(G_{21}^{a-1})$ and $t(G_{22}^n) = t(G_{22}^{a-1}) + p_a$. It follows that $t(G_{22}^n) - p_{max} \leqslant (\alpha - 1)t(G_{21}^{a-1}) \leqslant (\alpha - 1)t(G_{21}^n)$. Similar to Case 1, we have

$$\frac{C^*}{C^{HM2}} \leqslant \frac{t(G_{22}^n) - p_{max} + t(G_{21}^n) + t(G_1)}{t(G_1^n) + t(G_{21}^n)} \leqslant \frac{\alpha t(G_{21}^n) + t(G_1)}{t(G_1^n) + t(G_{21}^n)} \leqslant \alpha.$$

This contradicts the assumption that $\mathcal{J}$ is a minimum counter example. Therefore, job J_k is the last one assigned to machine M_2 by Algorithm $HM2$. □

By Lemma 5, we know that $G_{22}^n = \{J_B\} \cup \beta \cup \{J_k\}$, where β is the set of all the jobs except J_B assigned to machine M_2 before job J_k. In the rest of the proof, we show that the minimum counter example $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$, which contains job J_k, does not exist. Next, we find the total size of all the jobs in β and then we distinguish three possible subcases according to the number of jobs in G_{21}^n and the size of $t(\beta)$.

Lemmas 3 and 4 show that

$$p_k > \left(\alpha - 1 - \frac{1}{2\alpha - 3}\right)p_{max} + (\alpha - 1)t(\beta).$$

Combining with $p_k \leqslant p_{max}$, we have

$$t(\beta) < \frac{5 - 2\alpha}{2\alpha - 3} p_{max} \approx 0.01917 p_{max}. \tag{10}$$

Subcase 2.1. There is only one job in G_{21}^n.

At this point, we have

$$C^{HM2} = L_1 = t(G_1) + t\big(G_{21}^n\big), \qquad t\big(G_{22}^n\big) = p_{max} + t(\beta) + p_k.$$

Combining with inequalities (5) and (10), we have

$$t(G_1) + t\big(G_{21}^n\big) < \frac{1}{2\alpha - 1} t\big(G_{22}^n\big) \leqslant \frac{1}{2\alpha - 1}\big(2p_{max} + t(\beta)\big) < \frac{1}{2\alpha - 3} p_{max}.$$

It follows that $t(G_1) + t(G_{21}^n) + t(\beta) < \frac{1}{2\alpha-3} p_{max} + \frac{5-2\alpha}{2\alpha-3} p_{max} < 0.52875 p_{max}$. Therefore, in the optimal schedule, the only one job in G_{21}^n must be assigned together with one of the two jobs J_{max} and J_k. It is clear that

$$C^* \leqslant \min\big\{p_{max} + t(\beta) + t(G_1), t\big(G_{21}^n\big) + p_k + t(G_1)\big\} \leqslant p_{max} + t(\beta) + t(G_1).$$

Note that $C^* > \alpha C^{HM2}$. So we get $p_{max} + t(\beta) + t(G_1) > \alpha(t(G_1) + t(G_{21}^n))$. Combining with Lemma 4, we get $t(\beta) > (\alpha(\alpha - 2) - 1) p_{max} \approx 0.19394 p_{max}$. This contradicts inequality (10). A minimum counter example of this case does not exist.

Subcase 2.2. There are at least two jobs in G_{21}^n and $t(\beta) > 0$.

Since there are at least two jobs of class 2 assigned to machine M_1, let job $J_b = \{p_b, 2\}$ be the last one and δ be the set of all the jobs of class 2 assigned to machine M_1 before J_b. In this subcase, the job set $G_{22}^n = \{J_B\} \cup \beta \cup \{J_k\}$. Note that all the jobs in β are assigned to machine M_2 by Step 4.2, which yields

$$t(\delta) + t(\beta) > (\alpha - 2) p_{max} \approx 0.48119 p_{max}. \tag{11}$$

According to the definition of job J_b, we have $t(\beta) + p_b > (\alpha - 1) t(\delta)$; otherwise, job J_b must be assigned to machine M_2 by Step 4.2. It follows that

$$t(\delta) + t(\beta) + p_b > \alpha t(\delta). \tag{12}$$

According to inequalities (8) and (10), we have

$$t(\delta) + t(\beta) + p_b < \left(\frac{6 - 2\alpha}{2\alpha - 3}\right) p_{max}. \tag{13}$$

Combining with (12) and (13), we obtain $t(\delta) < \frac{6-2\alpha}{\alpha(2\alpha-3)} p_{max}$. This implies that

$$t(\delta) + t(\beta) < \frac{6 - 2\alpha}{\alpha(2\alpha - 3)} p_{max} + \frac{5 - 2\alpha}{2\alpha - 3} p_{max} \approx 0.23227 p_{max},$$

which contradicts inequality (11). A minimum counter example of this case does not exist.

Subcase 2.3. There are at least two jobs in G_{21}^n and $t(\beta) = 0$.

Similar to Subcase 2.2, let job $J_b = \{p_b, 2\}$ be the last one and δ be the set of all the jobs of class 2 assigned to machine M_1 before J_b. The equality $t(\beta) = 0$ shows that the job set $G_{22}^n = \{J_B\} \cup \{J_k\}$. In this subcase, we have $C^{HM2} = t(G_1) + t(\delta) + p_b \leqslant \frac{1}{2\alpha-3} p_{max}$. Next, we prove that $G_1 = \emptyset$ for a minimum counter example. Otherwise, if $G_1 \neq \emptyset$, let $\mathcal{J}' = \mathcal{J} \cup G_1$ be a minimum counter example, where all the jobs in $\mathcal{J}$ are of class 2. We now provide the relations between the competitive ratios of both job sequences $\mathcal{J}'$ and $\mathcal{J}$. Let $C^*(\mathcal{J}')$ and $C^*(\mathcal{J})$ be the optimal values in an offline version, and $C^{HM2}(\mathcal{J}')$ and $C^{HM2}(\mathcal{J})$ be the objective values produced by $HM2$ of $\mathcal{J}'$ and $\mathcal{J}$, respectively. It is clear that $C^{HM2}(\mathcal{J}') = C^{HM2}(\mathcal{J}) + t(G_1)$, and $C^*(\mathcal{J}') \leqslant C^*(\mathcal{J}) + t(G_1)$. Since $\mathcal{J}'$ is a counter example, we have

$$\frac{C^*(\mathcal{J}')}{C^{HM2}(\mathcal{J}')} > \alpha.$$

It follows that

$$\frac{C^*(\mathcal{J})}{C^{HM2}(\mathcal{J})} > \frac{C^*(\mathcal{J}) + t(G_1)}{C^{HM2}(\mathcal{J}) + t(G_1)} \geqslant \frac{C^*(\mathcal{J}')}{C^{HM2}(\mathcal{J}')} > \alpha.$$

This shows that instance $\mathcal{J}$ is also a counter example and the number of jobs in $\mathcal{J}$ is fewer than that in $\mathcal{J}'$ since $G_1 \neq \emptyset$. This contradicts the assumption that $\mathcal{J}'$ is a minimum counter example. Therefore, $G_1 = \emptyset$ in a minimum counter example.

Since $G_1 = \emptyset$, we have $C^{HM2} = L_1 = t(\delta) + p_b \leqslant \frac{1}{2\alpha-3} p_{max}$ by Lemma 4 and $t(G_{22}^n) = p_{max} + p_k \leqslant 2p_{max}$. In the optimal offline schedule, job J_b must be assigned to one machine, together with one of the two jobs J_B and J_k. Thus, $C^* \leqslant p_{max} + t(\delta)$. Since $C^*/C^{HM2} > \alpha$, we have

$$C^{HM2} = L_1 = t(\delta) + p_b < \frac{1}{\alpha}\big(p_{max} + t(\delta)\big). \tag{14}$$

Combining with Lemma 4, we have $\frac{1}{\alpha}(p_{max} + t(\delta)) > (\alpha - 2)p_{max}$ and it follows that

$$t(\delta) > \big(\alpha(\alpha - 2) - 1\big)p_{max}. \tag{15}$$

According to the definition of job J_b, we have $p_b > (\alpha - 1)t(\delta)$. Otherwise, $p_b + t(\beta) = p_b \leqslant (\alpha - 1)t(\delta) = (\alpha - 1)t(G_{21}^{b-1})$ and job J_b must be assigned to machine M_2 by Step 4.2. Inequality (14) shows that $p_b < \frac{1}{\alpha} p_{max} + (\frac{1}{\alpha} - 1)t(\delta)$. Therefore, $(\alpha - 1)t(\delta) < p_b < \frac{1}{\alpha} p_{max} + (\frac{1}{\alpha} - 1)t(\delta)$ and it follows that

$$t(\delta) \leqslant \frac{1}{\alpha^2 - 1} p_{max}. \tag{16}$$

Note that $\frac{1}{\alpha^2-1} = (\alpha(\alpha-2)-1)$ since α is the root of the equation $x^3 - 2x^2 - 2x + 2 = 0$. Inequalities (15) and (16) contradict each other. A minimum counter example of this case does not exist.

The above analysis confirms that there exists no such minimum counter example. Therefore, the competitive ratio of Algorithm *HM*2 is at most α. □

From Theorems 3 and 4, we know that *HM*2 is an optimal algorithm with a competitive ratio α for the case where we know $J_{max} = (p_{max}, 2)$ in advance.

4. Total size is known

In this section we design an optimal algorithm for the case where the total size of all the jobs $T = \sum_{1 \leqslant j \leqslant n} p_j$ is known in advance. We first provide a lower bound on the competitive ratio.

Theorem 5. *Any semi-online Algorithm A for the problem has a competitive ratio at least* 2.

Proof. First, we assume without loss of generality that the total size of all the jobs is 4, i.e., $T = 4$, and begin with job $J_1 = (1, 2)$. If job J_1 is scheduled on machine M_1, then we generate jobs $J_2 = (2, 1)$ and $J_3 = (1, 2)$. At this point, $C^* = 2$ and we have $C^A \leqslant 1$ since job J_2 must be scheduled on machine M_1, so $C^*/C^A \geqslant 2$. If job J_1 is scheduled on machine M_2, we generate job $J_2 = (2, 2)$. If job J_2 is scheduled on machine M_1, then we generate job $J_3 = (1, 1)$, which yields $C^*/C^A \geqslant 2$. Otherwise, if job J_2 is scheduled on machine M_2, then we generate job $J_3 = (1, 2)$. We have $C^* = 2$ and $C^A \leqslant 1$ no matter which machine job J_3 is assigned to. Thus the competitive ratio of any algorithm for this problem is at least 2. □

Algorithm *HS*.

1. Assign all the jobs of class 1 to machine M_1;
2. Always assign the current job of class 2 to machine M_2 until there exists a job $J_i = \{p_i, 2\}$ such that $t(G_{22}^{i-1}) + p_i > \frac{3T}{4}$;
   2.1 If $t(G_{22}^{i-1}) \geqslant \frac{T}{4}$, then assign J_i and all the remaining jobs of class 2 to M_1;
   2.2 else if $\frac{T-p_i}{2} \leqslant t(G_{22}^{i-1}) < \frac{T}{4}$, then assign J_i to machine M_1 and all the remaining jobs of class 2 to M_2;
   2.3 otherwise, $t(G_{22}^{i-1}) < \frac{T-p_i}{2} < \frac{T}{4}$, then assign J_i to machine M_2 and all the remaining jobs of class 2 to M_1.

Theorem 6. *The competitive ratio of Algorithm HS for the problem is at most* 2.

Proof. We distinguish two cases according to the final load of machine M_2.

Case 1. $L_2 = t(G_{22}^n) \leqslant \frac{3T}{4}$.

In this case, we have $L_1 = T - L_2 \geqslant \frac{T}{4}$. By Lemma 2, we have $C^* \leqslant \frac{T}{2}$. If we also have $L_2 \geqslant \frac{T}{4}$, then $C^{HS} = \min\{L_1, L_2\} \geqslant \frac{T}{4} \geqslant \frac{C^*}{2}$ and we are done.

Assume that $L_2 < \frac{T}{4}$. So we have $C^A = L_2$ and $L_1 = t(G_1) + t(G_{21}^n) > \frac{3T}{4}$. If $t(G_{21}^n) = 0$, then Algorithm *HS* gives an optimal schedule. If $t(G_{21}^n) \neq 0$, then there is at least one job in G_{21}^n. Let job $J_k = \{p_k, 2\}$ be the last one (may be the only one) of class 2 assigned to machine M_1. It is clear that $t(G_{22}^{k-1}) \leqslant L_2 < \frac{T}{4}$, which yields that job J_k is assigned to machine M_1 by Step 2.2 of Algorithm *HS*. Therefore, we have the following inequality

$$\frac{T - p_k}{2} \leqslant t\big(G_{22}^{k-1}\big) \leqslant L_2 < \frac{T}{4}. \tag{17}$$

It follows that $p_k > \frac{T}{2}$. In the optimal schedule, we must assign job J_k to machine M_2 alone and all the other jobs to machine M_1. Combining with inequality (17), we have

$$C^* = T - p_k \leqslant 2t\big(G_{22}^{k-1}\big) \leqslant 2L_2 = 2C^{HS}.$$

Case 2. $L_2 = t(G_{22}^n) > \frac{3T}{4}$.

In this case, we have $C^A = L_1 = t(G_1) + t(G_{21}^n) = T - L_2 < \frac{T}{4}$. Let job $J_k = \{p_k, 2\}$ be the last job assigned to machine M_2. It is clear that $t(G_{22}^n) = t(G_{22}^{k-1}) + p_k > \frac{3T}{4}$ and $t(G_{21}^{k-1}) \leqslant L_1 < \frac{T}{4}$. So job J_k is assigned to machine M_2 by Step 2.3 of Algorithm *HS*. The following inequality holds

$$t\big(G_{22}^{k-1}\big) \leqslant \frac{T - p_k}{2} < \frac{T}{4}. \tag{18}$$

It follows that $p_k > \frac{T}{2}$. In the optimal schedule, we must assign job J_k to machine M_2 alone and all the other jobs to machine M_1, so $C^* = T - p_k$. Combining with inequality (18), we have

$$C^{HS} = L_1 = T - t\big(G_{22}^n\big) = T - t\big(G_{22}^{k-1}\big) - p_k \geqslant \frac{T - p_k}{2} = \frac{C^*}{2}.$$

From the above analysis, we see that the competitive ratio of *HS* is at most 2. □

From Theorems 5 and 6, we know that *HS* is an optimal algorithm for the case where the total size of all the jobs is known in advance and its competitive ratio is 2.

Corollary 1. *For the case where we know both the total size of all the jobs and the largest size of the jobs in advance, Algorithm HS is also optimal with a competitive ratio* 2*.*

Proof. Assume that we know $T = 4$ and $p_{max} = 2$ in advance. Then we can prove that a lower bound for the problem is 2 by using the job sequences used in the proof of Theorem 5. Then Theorem 6 shows that Algorithm *HS* is also optimal for this problem with a competitive ratio 2. □

5. Conclusions

In this paper we study several cases of the semi-online version of the machine covering problem on two hierarchical parallel identical machines, where we know T, p_{max}, or J_{max} in advance. We provide optimal algorithms for all these cases. It is an interesting problem to design (optimal) algorithms for the case where there are $m > 2$ hierarchical parallel identical machines. To design optimal algorithms for the case where there are uniform machines with general speeds is another challenging topic for future research.

Acknowledgements

Wu was supported in part by Zhejiang Provincial Natural Science Foundation of China (Grant No. LQ13A010010) and Ningbo Natural Science Foundation (Grant No. 2012A610023). Ji was supported in part by the Humanities and Social Sciences Planning Foundation of the Ministry of Education (Grant No. 13YJA630034), the National Basic Research Program of China (973 Program) (Grant No. 2012CB315804), the Key Innovation Team of Science Technology Department of Zhejiang Province (Grant No. 2010R50041), and the Contemporary Business and Trade Research Center of Zhejiang Gongshang University, which is the Key Research Institute of Social Sciences and Humanities of the Ministry of Education of China.

References

[1] A. Bar-Noy, A. Freund, J. Naor, On-line load balancing in a hierarchical server topology, SIAM J. Comput. 31 (2001) 527–549.
[2] O. Chassid, L. Epstein, The hierarchical model for load balancing on two machines, J. Comb. Optim. 15 (2008) 305–314.
[3] T.C.E. Cheng, H. Kellerer, V. Kotov, Semi-on-line multiprocessor scheduling with given total processing time, Theoret. Comput. Sci. 337 (2005) 134–146.
[4] P. Crescenzi, G. Gambosi, P. Penna, On-line algorithms for the channel assignment problem in cellular networks, Discrete Appl. Math. 137 (3) (2004) 237–266.
[5] B.L. Deuermeyer, D.K. Friesen, M.A. Langston, Scheduling to maximize the minimum processor finish time in a multiprocessor system, SIAM J. Algebr. Discrete Methods 3 (1982) 190–196.
[6] C.A. Glass, H. Kellerer, Parallel machine scheduling with job assignment restrictions, Naval Res. Logist. 54 (2007) 250–257.
[7] H.C. Hwang, S.Y. Chang, K. Lee, Parallel machine scheduling under a grade of service provision, Comput. Oper. Res. 31 (2004) 2055–2061.

[8] M. Ji, T.C.E. Cheng, An FPTAS for parallel-machine scheduling under a grade of service provision to minimize makespan, Inform. Process. Lett. 108 (2008) 171–174.
[9] Y.W. Jiang, Online scheduling on parallel machines with two GoS levels, J. Comb. Optim. 16 (2008) 28–38.
[10] Y.W. Jiang, Y. He, C.M. Tang, Optimal online algorithms for scheduling on two identical machines under a grade of service, J. Zhejiang Univ. Sci. A 7 (3) (2006) 309–314.
[11] M. Liu, C.B. Chu, Y.F. Xu, F.F. Zheng, Semi-online scheduling on 2 machines under a grade of service provision with bounded processing times, J. Comb. Optim. 21 (2011) 138–149.
[12] J. Park, S.Y. Chang, K. Lee, Online and semi-online scheduling of two machines under a grade of service provision, Oper. Res. Lett. 34 (2006) 692–696.
[13] Z.Y. Tan, Y. Wu, Optimal semi-online algorithms for machine covering, Theoret. Comput. Sci. 372 (2007) 69–80.
[14] Y. Wu, M. Ji, Q.F. Yang, Optimal semi-online scheduling algorithms on two parallel identical machines under a grade of service provision, Int. J. Prod. Econ. 135 (2012) 367–371.
[15] A. Zhang, Y.W. Jiang, L.D. Fan, J.L. Hu, Optimal online algorithms on two hierarchical machines with tightly-grouped processing times, J. Comb. Optim. (2013), http://dx.doi.org/10.1007/s10878-013-9627-7.
[16] A. Zhang, Y.W. Jiang, Z.Y. Tan, Online parallel machines scheduling with two hierarchies, Theoret. Comput. Sci. 410 (2009) 3597–3605.