Theoretical Computer Science 412 (2011) 1092–1098

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Online and semi-online hierarchical scheduling for load balancing on uniform machines☆

Li-ying Hou, Liying Kang*

Department of Mathematics, Shanghai University, Shanghai 200444, PR China

ARTICLE INFO

Article history:
Received 11 October 2010
Received in revised form 23 November 2010
Accepted 3 December 2010
Communicated by editor D.-Z. Du

Keywords:
Scheduling
Online
Semi-online
Uniform machine
Hierarchy
Fractional assignment

ABSTRACT

In this paper, we consider online and semi-online hierarchical scheduling for load balancing on m parallel uniform machines with two hierarchies. There are k machines with speed s and hierarchy 1 which can process all the jobs, while the remaining $m-k$ machines with speed 1 and hierarchy 2 can only process jobs with hierarchy 2. For the model with the objective to maximize the minimum machine completion time, we show that no online algorithm can achieve bounded competitive ratio, i.e., there exists no competitive algorithm. We overcome this barrier by way of fractional assignment, where each job can be arbitrarily split between the machines. We design a best possible algorithm with competitive ratio $\frac{2ks+m-k}{ks+m-k}$ for any $0 < s < \infty$. For the model with the objective of minimizing makespan, we give a best possible algorithm with competitive ratio $\frac{(ks+m-k)^2}{k^2s^2+k(m-k)s+(m-k)^2}$ for any $0 < s < \infty$. For the semi-online model with the objective to minimize makespan, where the total job size (*sum*) is known in advance, we present an optimal algorithm (i.e., an algorithm of competitive ratio 1).

1. Introduction

Problem statement. The hierarchical scheduling problem is an important practical paradigm. It has many applications in diverse areas, such as assigning classes of service to calls in wireless communication networks, routing queries to hierarchical databases, signing documents by ranking executives, and upgrading classes of cars by car rental companies. Consider a wireless communication network, similar to that of cellular phone system, where customers arrive (i.e., turn on their phone) one by one in an arbitrary order. Upon arrival each customer discloses the extent of service it requires and must be assigned a base station, from those within range, to service it. Improper station assignment may cause overloading of some station (or equivalently, entail penalty of hand-off). Thus it is desirable to spread the load as evenly as possible.

In this paper, we study online and semi-online hierarchical scheduling for load balancing on m parallel uniform machines with two hierarchies. Given a set of jobs $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ and a set of machines $\mathcal{M} = \{M_1, M_2, \ldots, M_m\}$, jobs arrive one by one. The job J_j has a positive size p_j and a hierarchy $g_j = 1$ or 2, $j = 1, \ldots, n$, and the machine M_i is also associated with a hierarchy $g(M_i)$, $i = 1, \ldots, m$. The job J_j can be scheduled on the machine M_i only if $g_j \geq g(M_i)$. There are k machines $\mathcal{M}_1 = \{M_1, \ldots, M_k\}$ with speed $s (0 < s < \infty)$, the time required to process J_j is $\frac{p_j}{s}$. The remaining $m-k$ machines $\mathcal{M}_2 = \{M_{k+1}, \ldots, M_m\}$ have speed 1, and the time required to process J_j is p_j. We assume that $g(M_i) = 1, i = 1, \ldots, k$ and $g(M_i) = 2, i = k+1, \ldots, m$, where $1 \leq k \leq m-1$. According to the scheduling notation introduced by Graham et al. [7], the problem is denoted by $Qm|online, g = 2, frac|C_{min}(C_{max})$.

☆ Research was partially supported by the National Nature Science Foundation of China (No. 10971131) and Shanghai Leading Academic Discipline Project (No. S30104).

* Corresponding author. Tel.: +86 21 66135652.
E-mail address: lykang@shu.edu.cn (L. Kang).

0304-3975/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2010.12.008

For online scheduling problems, we measure the performance of online algorithms using the competitive ratio. For an online algorithm A, let C^A denote the cost of A, and C^{OPT} denote the cost of an optimal offline algorithm. For minimization problems, the competitive ratio of A is the infimum $\mathcal{R} \geq 1$ such that for any input, $C^A \leq \mathcal{R}C^{OPT}$. For maximization problems, the competitive ratio of A is the infimum $\mathcal{R} \geq 1$ such that for any input, $C^{OPT} \leq \mathcal{R}C^A$. An online algorithm A with competitive ratio c is called c-competitive algorithm. If the competitive ratio of an online algorithm is at most c, we say that it is c-competitive. If no $\mathcal{R}$ satisfying the inequality exists, we say that the competitive ratio is unbounded or ∞.

Related works. Online load balancing on uniform machines was first studied by Aspnes et al. [1] and Berman et al. [3]. Recently, Chassid and Epstein [4] considered the hierarchical model on two uniform machines, they showed that no online algorithm can achieve bounded competitive ratio for the objective to maximize the minimum machine completion time. For $Q2|online, g = 2, frac|C_{min}$, they proposed a best possible algorithm with competitive ratio $\frac{2s+1}{s+1}$ for any $0 < s < \infty$. For $Q2|online, g = 2, sum|C_{min}$, i.e., the semi-online model where the total job size (sum) is known in advance, they gave a best possible algorithm with competitive ratio $\max\{s + \frac{1}{s}, 1 + \frac{1}{s}\}$ for any $0 < s < \infty$. For $Q2|online, g = 2, frac, sum|C_{min}(C_{max})$, they presented an optimal algorithm. For $Q2|online, g = 2, frac|C_{max}$, they provided a best possible algorithm with competitive ratio $\frac{(1+s)^2}{1+s+s^2}$ for any $0 < s < \infty$. For $Pm|online, g = 2, frac|C_{max}$, Zhang et al. [13] proposed a best possible algorithm with competitive ratio $\frac{m^2}{m^2-km+k^2}$.

Dosa and Epstein [6] studied the preemptive version $Q2|online, g = 2, pmtn|C_{max}$, they obtained a best possible algorithm with competitive ratio $\max\{\frac{(1+s)^2}{1+s+s^2}, \frac{s(1+s)^2}{1+s^2+s^3}\}$. And later, Tan and Zhang [12] considered the model $Q2|online, g = 2|C_{max}$, they proposed a best possible algorithm with competitive ratio $\min\{1+s, 1+\frac{1+s}{1+s+s^2}\}$ for $0 < s < 1$, and gave a best possible algorithm with competitive ratio $\min\{\frac{1+s}{s}, 1 + \frac{2s}{1+s+s^2}\}$ for $s \geq 1$.

For $Pm|online, g = m|C_{max}$, Bar-Noy et al. [2] and Crescenzi et al. [5] independently designed a non-preemptive $e + 1$-competitive algorithm, which is also showed to be e-competitive when all jobs have unit size. Afterwards, Jiang [10] considered $Pm|online, g = 2|C_{max}$, he proposed two online algorithms and a lower bound. Recently, Zhang et al. [14] further studied $Pm|online, g = 2|C_{max}$, they presented an online algorithm with competitive ratio $1 + \frac{m^2-m}{m^2-km+k^2}$. For $P2|online, g = 2|C_{max}$, Jiang et al. [9] and Park et al. [11] independently presented a best possible algorithm with competitive ratio $\frac{5}{3}$. In addition, Hwang et al. [8] considered the offline version of the problem and presented an algorithm with performance ratio no more than $\frac{5}{4}$ for $m = 2$ and $2 - \frac{1}{m-1}$ for $m \geq 3$.

Our results. In this paper, we consider $Qm|online, g = 2, frac|C_{min}(C_{max})$. For $Qm|online, g = 2|C_{min}$, we show that no online algorithm can achieve bounded competitive ratio. In order to overcome this barrier, we consider the fractional assignment model $Qm|online, g = 2, frac|C_{min}$, where each job can be arbitrarily split between the machines. We design a best possible algorithm with competitive ratio $\frac{2ks+m-k}{ks+m-k}$ for any $0 < s < \infty$. For $Qm|online, g = 2, frac|C_{max}$, we design a best possible algorithm with competitive ratio $\frac{(ks+m-k)^2}{k^2s^2+k(m-k)s+(m-k)^2}$ for any $0 < s < \infty$. For $Qm|online, g = 2, frac, sum|C_{max}$, i.e., the semi-online model where the total job size (sum) is known in advance, we give an optimal algorithm with competitive ratio 1.

This paper is organized as follows. In Section 2, we give some basic notations and a useful lemma, and show that no online algorithm can achieve bounded competitive ratio for $Qm|online, g = 2|C_{min}$. In Sections 3 and 4, we consider the model $Qm|online, g = 2, frac|C_{min}$ and $Qm|online, g = 2, frac|C_{max}$, respectively. In Section 5, we present an optimal algorithm for $Qm|online, g = 2, frac, sum|C_{max}$. Finally some conclusions are given in Section 6.

2. Preliminaries

The following notations and the lemma are required in this paper.

T	The total size of all jobs.
T_i	The total size of jobs with hierarchy i.
P_i	The total size of jobs assigned to M_i.
L_i^{j-1}	The current load of M_i right before the assignment of job J_j.
L_i	The final load of machine M_i.

Next we give the bounds on C^{OPT}.

Lemma 2.1. *For the offline problem $Qm|g = 2, frac|C_{max}$, we have $C^{OPT} \geq \max\{\frac{T_1}{ks}, \frac{T}{ks+m-k}\}$. For the offline problem $Qm|g = 2, frac|C_{min}$, we have $C^{OPT} \leq \min\{\frac{T_2}{m-k}, \frac{T}{ks+m-k}\}$.*

Proof. For the offline problem $Qm|g = 2, frac|C_{max}$, it is clear that $C^{OPT} \geq \frac{T}{ks+m-k}$. Note that all jobs with hierarchy 1 can be only processed on machines in $\mathcal{M}_1$, which implies that $C^{OPT} \geq \frac{T_1}{ks}$. Hence the result follows. For the offline problem $Qm|g = 2, frac|C_{min}$, it is clear that $C^{OPT} \leq \frac{T}{ks+m-k}$. Note that all jobs with hierarchy 2 can be processed on all machines in $\mathcal{M}_1$ and $\mathcal{M}_2$, then $C^{OPT} \leq \frac{T_2}{m-k}$. So the result holds. □

In the following, we show that no online algorithm with bounded competitive ratio exists for $Qm|online, g = 2|C_{min}$.

Theorem 2.1. *For $Qm|online, g = 2|C_{min}$, any online algorithm has an unbounded competitive ratio.*

Proof. Let $N > 0$ be a sufficiently large number. Consider an online algorithm A and the following sequence of jobs.

The first $m - k$ jobs are of size 1 and hierarchy 2. If some of these jobs are scheduled on machines in $\mathcal{M}_1$ or if there is a machine in $\mathcal{M}_2$ which processes at least two jobs, then k jobs with size s and hierarchy 1 arrive. For both cases, we have $C^A = 0$, $C^{OPT} = 1$, and thus $R = \infty$.

If the first $m - k$ jobs are all scheduled on machines in $\mathcal{M}_2$ and each machine in $\mathcal{M}_2$ process exactly one job, then $m - k$ jobs with size sN and hierarchy 2 arrive. If some machine in $\mathcal{M}_1$ does not process any job, then no new job arrives. We have $C^A = 0$, $C^{OPT} \geq 1$, $R = \infty$ again. If each machine in $\mathcal{M}_1$ processes at least one job, then the last k jobs with size $s(sN + 1)$ and hierarchy 1 arrive. Obviously, the last k jobs must be scheduled on machines in $\mathcal{M}_1$. Hence $C^A = 1$, $C^{OPT} = sN + 1$, and $R = \frac{C^{OPT}}{C^A} = sN + 1 \to \infty (N \to \infty)$. □

3. The model $Qm|online, g = 2, frac|C_{min}$

The next algorithm is defined for any $0 < s < \infty$.

Algorithm A1. When a new job J_j arrives:

1. If $g_j = 1$, split job J_j into k equal parts. Each machine in $\mathcal{M}_1$ processes a part of size $\frac{p_j}{k}$ of job J_j.
2. If $g_j = 2$, split job J_j into m parts in the ratio

$$\underbrace{(m-k)s : (m-k)s : \ldots : (m-k)s}_{\text{k}} : \underbrace{ks + m - k : ks + m - k : \ldots : ks + m - k}_{\text{m-k}}.$$

Each machine in $\mathcal{M}_1$ processes a part of size $\frac{sp_j}{2ks+m-k}$ of job J_j, and each machine in $\mathcal{M}_2$ processes a part of size $\frac{(ks+m-k)p_j}{2k(m-k)s+(m-k)^2}$ of job J_j.

Clearly, for the jobs with hierarchy 1, the best way is to schedule them on machines in $\mathcal{M}_1$ evenly. The reason that we split the jobs with hierarchy 2 into m parts and assign a part of the job to each of the machines in $\mathcal{M}_1$ is to ensure that Algorithm A1 can get a better competitive ratio when there are no jobs with hierarchy 1. Since machines in $\mathcal{M}_1$ and $\mathcal{M}_2$ are the same, respectively, the part assigned to machines in $\mathcal{M}_1$ and $\mathcal{M}_2$ are equal, respectively.

From the designment of Algorithm A1, the total size of jobs on machines in $\mathcal{M}_1$ (or $\mathcal{M}_2$) are equal, and the loads of machines in $\mathcal{M}_1$ (or $\mathcal{M}_2$) are equal when Algorithm A1 is completed. Let $\mathcal{P}^1 = P_1 = P_2 = \cdots = P_k$, $\mathcal{P}^2 = P_{k+1} = P_{k+2} = \cdots = P_m$, $\mathcal{L}^1 = L_1 = L_2 = \cdots = L_k$ and $\mathcal{L}^2 = L_{k+1} = L_{k+2} = \cdots = L_m$. Thus, $\mathcal{L}^1 = \frac{\mathcal{P}^1}{s}$ and $\mathcal{L}^2 = \mathcal{P}^2$.

Theorem 3.1. *Algorithm A1 has a competitive ratio of at most $\frac{2ks+m-k}{ks+m-k}$ for any $0 < s < \infty$.*

Proof. If there is no job with hierarchy 2, then $C^{A1} = 0$ and $C^{OPT} = 0$, thus the competitive ratio is 1. Otherwise, we have

$$\mathcal{L}^2 = \mathcal{P}^2 = \frac{ks + m - k}{2k(m-k)s + (m-k)^2}(T - T_1),$$

and

$$\begin{aligned}\mathcal{P}^1 &= \frac{T_1}{k} + \frac{s}{2ks + m - k}(T - T_1)\\ &= \frac{s}{2ks + m - k}T + \frac{ks + m - k}{k(2ks + m - k)}T_1,\end{aligned}$$

and thus

$$\mathcal{L}^1 = \frac{1}{2ks + m - k}T + \frac{ks + m - k}{ks(2ks + m - k)}T_1.$$

If $T_1 \leq \frac{ks}{ks+m-k}T$ and $C^{A1} = \mathcal{L}^1$, then by Lemma 2.1 and $T_1 \geq 0$, we have

$$\begin{aligned}\frac{C^{OPT}}{C^{A1}} &\leq \frac{\frac{T}{ks+m-k}}{\frac{1}{2ks+m-k}T + \frac{ks+m-k}{ks(2ks+m-k)}T_1}\\ &\leq \frac{\frac{T}{ks+m-k}}{\frac{T}{2ks+m-k}}\\ &= \frac{2ks + m - k}{ks + m - k}.\end{aligned}$$

If $T_1 \le \frac{ks}{ks+m-k}T$ and $C^{A1} = \mathcal{L}^2$, then by Lemma 2.1 we have

$$\begin{aligned}\frac{C^{OPT}}{C^{A1}} &\le \frac{\frac{T}{ks+m-k}}{\frac{ks+m-k}{2k(m-k)s+(m-k)^2}(T-T_1)}\\ &\le \frac{\frac{T}{ks+m-k}}{\frac{ks+m-k}{2k(m-k)s+(m-k)^2}\cdot\frac{m-k}{ks+m-k}T}\\ &= \frac{2ks+m-k}{ks+m-k}.\end{aligned}$$

If $T_1 \in (\frac{ks}{ks+m-k}T, T]$, then $\mathcal{L}^1 > \frac{\frac{ks}{ks+m-k}T}{ks} = \frac{1}{ks+m-k}T$. Hence,

$$\begin{aligned}\mathcal{L}^2 &= \frac{ks+m-k}{2k(m-k)s+(m-k)^2}(T-T_1)\\ &< \frac{1}{m-k}(T-T_1)\\ &< \frac{1}{m-k}\left(T-\frac{ks}{ks+m-k}T\right)\\ &= \frac{1}{ks+m-k}T\\ &< \mathcal{L}^1.\end{aligned}$$

So, $C^{A1} = \mathcal{L}^2$. By Lemma 2.1 we have

$$\begin{aligned}\frac{C^{OPT}}{C^{A1}} &\le \frac{\frac{T-T_1}{m-k}}{\frac{ks+m-k}{2k(m-k)s+(m-k)^2}(T-T_1)}\\ &= \frac{2ks+m-k}{ks+m-k}.\end{aligned}$$ □

Theorem 3.2. *For $Qm|online, g = 2, frac|C_{min}$, any online algorithm has a competitive ratio of at least $\frac{2ks+m-k}{ks+m-k}$ for any $0 < s < \infty$.*

Proof. The job J_1 with $p_1 = m-k$ and $g_1 = 2$ arrives first. Let δ be the minimum part of J_1 scheduled on machines in $\mathcal{M}_2$.

If $\delta \ge \frac{ks+m-k}{2ks+m-k}$, then no new job arrives. At this time, $C^{OPT} = \frac{m-k}{ks+m-k}$. Let $\gamma = \min\{L_i | i = 1, 2, \ldots, k\}$, then

$$\begin{aligned}\gamma &\le \frac{(m-k)(1-\delta)}{ks}\\ &\le \frac{(m-k)\frac{ks}{2ks+m-k}}{ks}\\ &= \frac{m-k}{2ks+m-k}.\end{aligned}$$

So, $C^A \le \frac{m-k}{2ks+m-k}$. Hence

$$\frac{C^{OPT}}{C^A} \ge \frac{\frac{m-k}{ks+m-k}}{\frac{m-k}{2ks+m-k}} = \frac{2ks+m-k}{ks+m-k}.$$

If $\delta < \frac{ks+m-k}{2ks+m-k}$, then the last job J_2 with $p_2 = ks$ and $g_2 = 1$ arrives. Then $C^{OPT} = 1$. Since J_2 must schedule on machines in $\mathcal{M}_1$, $C^A \le \delta$. Hence

$$\frac{C^{OPT}}{C^A} \ge \frac{1}{\delta} \ge \frac{2ks+m-k}{ks+m-k}.$$ □

From Theorems 3.1 and 3.2, we obtain that Algorithm A1 is the best possible algorithm for $Qm|online, g = 2, frac|C_{min}$.

Corollary 3.1 ([4]). *If $k = 1, m = 2$, the problem $Qm|online, g = 2, frac|C_{min}$ is identical to $Q2|online, g = 2, frac|C_{min}$. Algorithm A1 has a competitive ratio of $\frac{2s+1}{s+1}$ for any $0 < s < \infty$, which is the best possible.*

4. The model $Qm|online, g = 2, frac|C_{max}$

The next algorithm is defined for any $0 < s < \infty$.

Algorithm A2. When a new job J_j arrives:

1. If $g_j = 1$, split job J_j into k equal parts. Each machine in $\mathcal{M}_1$ processes a part of size $\frac{p_j}{k}$ of job J_j.
2. If $g_j = 2$, split job J_j into m parts in the ratio

$$\underbrace{ks^2 : ks^2 : \ldots : ks^2}_{\text{k}} : \underbrace{ks + m - k : ks + m - k : \ldots : ks + m - k}_{\text{m-k}}.$$

Each machine in $\mathcal{M}_1$ processes a part of size $\frac{ks^2 p_j}{k^2s^2+k(m-k)s+(m-k)^2}$ of job J_j, and each machine in $\mathcal{M}_2$ processes a part of size $\frac{(ks+m-k)p_j}{k^2s^2+k(m-k)s+(m-k)^2}$ of job J_j.

The main idea of Algorithm A2 is similar to Algorithm A1. From the designment of Algorithm A2, the total size of jobs on machines in $\mathcal{M}_1$ (or $\mathcal{M}_2$) are equal, and the loads of machines in $\mathcal{M}_1$ (or $\mathcal{M}_2$) are equal when Algorithm A2 is completed. Let $\mathcal{P}^1 = P_1 = P_2 = \cdots = P_k$, $\mathcal{P}^2 = P_{k+1} = P_{k+2} = \cdots = P_m$, $\mathcal{L}^1 = L_1 = L_2 = \cdots = L_k$ and $\mathcal{L}^2 = L_{k+1} = L_{k+2} = \cdots = L_m$. Thus, $\mathcal{L}^1 = \frac{\mathcal{P}^1}{s}$ and $\mathcal{L}^2 = \mathcal{P}^2$.

Theorem 4.1. *Algorithm A2 has a competitive ratio of at most* $\frac{(ks+m-k)^2}{k^2s^2+k(m-k)s+(m-k)^2}$ *for any* $0 < s < \infty$.

Proof. By Algorithm A2, we have

$$\mathcal{L}^2 = \mathcal{P}^2 = \frac{ks + m - k}{k^2s^2 + k(m-k)s + (m-k)^2}(T - T_1),$$

and

$$\begin{aligned}\mathcal{P}^1 &= \frac{T_1}{k} + \frac{ks^2}{k^2s^2 + k(m-k)s + (m-k)^2}(T - T_1)\\ &= \frac{ks^2}{k^2s^2 + k(m-k)s + (m-k)^2}T + \frac{(m-k)(ks+m-k)}{k(k^2s^2 + k(m-k)s + (m-k)^2)}T_1,\end{aligned}$$

and thus

$$\mathcal{L}^1 = \frac{ks}{k^2s^2 + k(m-k)s + (m-k)^2}T + \frac{(m-k)(ks+m-k)}{ks(k^2s^2 + k(m-k)s + (m-k)^2)}T_1.$$

If $T_1 \le \frac{ks}{ks+m-k}T$ and $C^{A2} = \mathcal{L}^1$, then by Lemma 2.1 we have

$$\begin{aligned}\frac{C^{A2}}{C^{OPT}} &\le \frac{\frac{ks}{k^2s^2+k(m-k)s+(m-k)^2}T + \frac{(m-k)(ks+m-k)}{ks(k^2s^2+k(m-k)s+(m-k)^2)}T_1}{\frac{T}{ks+m-k}}\\ &\le \frac{\frac{ks}{k^2s^2+k(m-k)s+(m-k)^2}T + \frac{m-k}{k^2s^2+k(m-k)s+(m-k)^2}T}{\frac{T}{ks+m-k}}\\ &= \frac{(ks+m-k)^2}{k^2s^2 + k(m-k)s + (m-k)^2}.\end{aligned}$$

If $T_1 \le \frac{ks}{ks+m-k}T$ and $C^{A2} = \mathcal{L}^2$, then by Lemma 2.1 we have

$$\begin{aligned}\frac{C^{A2}}{C^{OPT}} &\le \frac{\frac{ks+m-k}{k^2s^2+k(m-k)s+(m-k)^2}(T - T_1)}{\frac{T}{ks+m-k}}\\ &\le \frac{(ks+m-k)^2}{k^2s^2 + k(m-k)s + (m-k)^2}.\end{aligned}$$

If $T_1 \in (\frac{ks}{ks+m-k}T, T]$, then $\mathcal{L}^2 \le \frac{T-T_1}{m-k} < \frac{\frac{m-k}{ks+m-k}T}{m-k} = \frac{1}{ks+m-k}T$. Hence,

$$\begin{aligned}\mathcal{L}^1 &= \frac{ks}{k^2s^2 + k(m-k)s + (m-k)^2}T + \frac{(m-k)(ks+m-k)}{ks(k^2s^2 + k(m-k)s + (m-k)^2)}T_1\\ &> \frac{ks}{k^2s^2 + k(m-k)s + (m-k)^2}T + \frac{m-k}{k^2s^2 + k(m-k)s + (m-k)^2}T\end{aligned}$$

$$= \frac{ks+m-k}{k^2s^2+k(m-k)s+(m-k)^2}T$$
$$> \frac{1}{ks+m-k}T$$
$$> \mathcal{L}^2.$$

So $C^{A2} = \mathcal{L}^1$. By Lemma 2.1, we have

$$\frac{C^{A2}}{C^{OPT}} \le \frac{\frac{ks}{k^2s^2+k(m-k)s+(m-k)^2}T + \frac{(m-k)(ks+m-k)}{ks(k^2s^2+k(m-k)s+(m-k)^2)}T_1}{\frac{T_1}{ks}}$$
$$< \frac{\frac{ks+m-k}{k^2s^2+k(m-k)s+(m-k)^2}T_1 + \frac{(m-k)(ks+m-k)}{ks(k^2s^2+k(m-k)s+(m-k)^2)}T_1}{\frac{T_1}{ks}}$$
$$= \frac{\frac{(ks+m-k)^2}{ks(k^2s^2+k(m-k)s+(m-k)^2)}T_1}{\frac{T_1}{ks}}$$
$$= \frac{(ks+m-k)^2}{k^2s^2+k(m-k)s+(m-k)^2}. \quad \square$$

Theorem 4.2. *For $Qm|online, g = 2, frac|C_{max}$, any online algorithm has a competitive ratio of at least $\frac{(ks+m-k)^2}{k^2s^2+k(m-k)s+(m-k)^2}$ for any $0 < s < \infty$.*

Proof. The job J_1 with $p_1 = m-k$ and $g_1 = 2$ arrives first. Let λ be the maximum part of J_1 scheduled on machines in $\mathcal{M}_2$.

If $\lambda \ge \frac{(m-k)(ks+m-k)}{k^2s^2+k(m-k)s+(m-k)^2}$, then no new job arrives. At this time, we have $C^{OPT} = \frac{m-k}{ks+m-k}$ and $C^A \ge \lambda$. Hence

$$\frac{C^A}{C^{OPT}} \ge \frac{\frac{(m-k)(ks+m-k)}{k^2s^2+k(m-k)s+(m-k)^2}}{\frac{m-k}{ks+m-k}}$$
$$= \frac{(ks+m-k)^2}{k^2s^2+k(m-k)s+(m-k)^2}.$$

If $\lambda < \frac{(m-k)(ks+m-k)}{k^2s^2+k(m-k)s+(m-k)^2}$, then the last job J_2 with $p_2 = ks$ and $g_2 = 1$ arrives. Obviously, J_2 must be scheduled on machines in $\mathcal{M}_1$. Then $C^{OPT} = 1$. Let $\mu = \max\{L_i | i = 1, \ldots, k\}$, then we have

$$\mu \ge \frac{(m-k)(1-\lambda)+ks}{ks}$$
$$> \frac{\frac{k^2(m-k)s^2}{k^2s^2+k(m-k)s+(m-k)^2}+ks}{ks}$$
$$= \frac{k(m-k)s}{k^2s^2+k(m-k)s+(m-k)^2}+1$$
$$= \frac{(ks+m-k)^2}{k^2s^2+k(m-k)s+(m-k)^2}.$$

Hence

$$\frac{C^A}{C^{OPT}} \ge \mu > \frac{(ks+m-k)^2}{k^2s^2+k(m-k)s+(m-k)^2}. \quad \square$$

From Theorems 4.1 and 4.2, we obtain that Algorithm A2 is the best possible algorithm for $Qm|online, g = 2, frac|C_{max}$.

Corollary 4.1 ([13]). *If $s = 1$, the problem $Qm|online, g = 2, frac|C_{max}$ is identical to $Pm|online, g = 2, frac|C_{max}$. Algorithm A2 has a competitive ratio of $\frac{m^2}{m^2-km+k^2}$, which is the best possible.*

Corollary 4.2 ([4]). *If $k = 1, m = 2$, the problem $Qm|online, g = 2, frac|C_{max}$ is identical to $Q2|online, g = 2, frac|C_{max}$. Algorithm A2 has a competitive ratio of $\frac{(1+s)^2}{1+s+s^2}$ for any $0 < s < \infty$, which is the best possible.*

5. The model $Qm|online, g = 2, frac, sum|C_{max}$

In this section we give an optimal algorithm which has competitive ratio 1.

Let $KL^{j-1} = \min\{L_i^{j-1} | i = k+1, \ldots, m\}$ and $M_{t_{j-1}}$ be one of the machines whose loads are KL^{j-1}, $t_{j-1} \in \{k+1, \ldots, m\}$.

Algorithm A3. 1. Let $j = 1$ and $L_i^0 = 0$, $i = 1, \ldots, m$.

2. If $g_j = 1$, assign job J_j by the following rule. Split job J_j into k equal parts, each machine in $\mathcal{M}_1$ processes a part of size $\frac{p_j}{k}$ of job J_j. Let $L_i^j = L_i^{j-1}$, $i = k+1, \ldots, m$, go to step 4.
3. If $g_j = 2$:
   (3.1) If $L_{k+1}^{j-1} = \cdots = L_m^{j-1} = \frac{T}{ks+m-k}$, assign job J_j by the rule in step 2. Let $L_{k+1}^j = \cdots = L_m^j = \frac{T}{ks+m-k}$, go to step 4.
   (3.2) Otherwise if $KL^{j-1} + p_j \leq \frac{T}{ks+m-k}$, assign job J_j to $M_{t_{j-1}}$. Let $L_{t_{j-1}}^j = L_{t_{j-1}}^{j-1} + p_j$ and $L_i^j = L_i^{j-1}$, $i \neq t_{j-1}$, $i \in \{k+1, \ldots, m\}$. Compute KL^j and determine M_{t_j}, go to step 4.
   (3.3) If $KL^{j-1} + p_j > \frac{T}{ks+m-k}$, assign a part of size $\frac{T}{ks+m-k} - KL^{j-1}$ of job J_j to $M_{t_{j-1}}$. Let $L_{t_{j-1}}^j = \frac{T}{ks+m-k}$ and $L_i^j = L_i^{j-1}$, $i \neq t_{j-1}$, $i \in \{k+1, \ldots, m\}$. Compute KL^j and determine M_{t_j}. Let $j = j+1$, $p_j = p_j - (\frac{T}{ks+m-k} - KL^{j-1})$, go to step 3.
4. If no new job arrives, stop. Otherwise, let $j = j+1$, go to step 2.

The main idea of Algorithm A3 is that the jobs with hierarchy 1 are assigned to machines in $\mathcal{M}_1$ evenly, and the jobs with hierarchy 2 are assigned to machines in $\mathcal{M}_2$ as many as possible, in other words, they will not be assigned to machines in $\mathcal{M}_1$ unless the loads of all the machines in $\mathcal{M}_2$ reach the threshold number.

Theorem 5.1. *The running time of Algorithm A3 is $O(nm^2)$.*

Proof. At each time, the machine with minimum load can be found in $O(m)$. Each job is split into at most m parts, and there are n jobs. Hence Algorithm A3 can be implemented with $O(nm^2)$. □

Theorem 5.2. *Algorithm A3 is an optimal algorithm.*

Proof. Let $\mathcal{L}^1 = \max\{L_1, \ldots, L_k\}$ and $\mathcal{L}^2 = \max\{L_{k+1}, \ldots, L_m\}$. By the designment of Algorithm A3, we have $L_i \leq \frac{T}{ks+m-k}$, $i = k+1, \ldots, m$. If $L_{k+1} = \cdots = L_m = \frac{T}{ks+m-k}$, then $L_1 = \cdots = L_m = \frac{T}{ks+m-k}$. Clearly, Algorithm A3 is optimal in this case. Otherwise, there is some machine M_i, $i \in \{k+1, \ldots, m\}$, such that $L_i < \frac{T}{ks+m-k}$. Then $\mathcal{L}^2 \leq \frac{T}{ks+m-k} < \mathcal{L}^1$. By Algorithm A3, all jobs with hierarchy 2 are scheduled on machines in $\mathcal{M}_2$ and all jobs with hierarchy 1 are scheduled on machines in $\mathcal{M}_1$, so Algorithm A3 is optimal as well. □

6. Conclusions

This paper studied online and semi-online hierarchical scheduling for load balancing on m parallel uniform machines with two hierarchies. There are k machines with speed s and hierarchy 1 which can process all the jobs, while the remaining $m-k$ machines with speed 1 and hierarchy 2 can only process jobs with hierarchy 2. We showed that no online algorithm can achieve bounded competitive ratio for the model with the objective to maximize the minimum machine completion time. We overcame this barrier by way of fractional assignment, where each job can be arbitrarily split between the machines. We designed the best possible algorithm with competitive ratio $\frac{2ks+m-k}{ks+m-k}$ for any $0 < s < \infty$. For the model with the objective of minimizing makespan, we gave the best possible algorithm with competitive ratio $\frac{(ks+m-k)^2}{k^2s^2+k(m-k)s+(m-k)^2}$ for any $0 < s < \infty$. For the semi-online model with the objective to minimize makespan, where the total job size (sum) is known in advance, we presented an optimal algorithm. Obtaining the best possible algorithms for m uniform machines with general speed and hierarchy is still an open problem.

References

[1] J. Aspnes, Y. Azar, A. Fiat, S. Plotkin, O. Waarts, On-line load balancing with applications to machine scheduling and virtual circuit routing, Journal of the ACM 44 (1997) 486–504.
[2] A. Bar-Noy, A. Freund, J. Naor, On-line load balancing in a hierarchical server topology, SIAM Journal on Computing 31 (2001) 527–549.
[3] P. Berman, M. Charikar, M. Karpinski, On-line load balancing for related machines, Jounal of Algorithms 35 (2000) 108–121.
[4] O. Chassid, L. Epstein, The hierarchical model for load balancing on two machines, Journal of Combinatorial Optimization 15 (2008) 305–314.
[5] P. Crescenzi, G. Gambosi, P. Penna, On-line algorithms for the channel assignment problem in cellular networks, Discrete Applied Mathematics 137 (2004) 237–266.
[6] G. Dosa, L. Epstein, Preemptive scheduling on a small number of hierarchical machines, Information and Computation 206 (2008) 602–619.
[7] R.L. Graham, E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, Optimization and approximation in deterministic sequencing and scheduling: a survey, Annals of Discrete Mathematics 5 (1979) 287–326.
[8] H. Hwang, S.Y. Chang, K. Lee, Parallel machine scheduling under a grade of service provition, Computers & Operations Research 31 (2004) 2055–2061.
[9] Y.W. Jiang, Y. He, C.M. Tang, Optimal online algorithms for scheduling on two identical machines under a grade of service, Journal of Zhejiang University Science 7A (2006) 309–314.
[10] Y.W. Jiang, Online scheduling on parallel machines with two GoS levels, Journal of Combinatorial Optimization 16 (2008) 28–38.
[11] J. Park, S.Y. Chang, K. Lee, Online and semi-online scheduling of two machines under a grade of service provision, Operations Research Letters 34 (2006) 692–696.
[12] Z.Y. Tan, A. Zhang, A note on hierarchical scheduling on two uniform machines, Journal of Combinatorial Optimization 20 (2010) 85–95.
[13] A. Zhang, Y.W. Jiang, Z.Y. Tan, Optimal algorithms for online hierarchical scheduling on parallel machines, Technical report, Zhejiang University (2008).
[14] A. Zhang, Y.W. Jiang, Z.Y. Tan, Online parallel machines scheduling with two hierarchies, Theoretical Computer Science 410 (2009) 3597–3605.