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:

Given $\mathcal{J}$, 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:

Given $\mathcal{J}$ and T (or p_{max}, or J_{max}), 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.