Information Processing Letters 113 (2013) 127–131

Contents lists available at SciVerse ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

Semi-online hierarchical scheduling problems with buffer or rearrangements

Xin Chen [a], Zhenzhen Xu [a], György Dósa [b], Xin Han [a,*], He Jiang [a]

[a] Software School, Dalian University of Technology, China
[b] Department of Mathematics, University of Pannonia, Veszprém, Hungary

ARTICLE INFO

Article history:
Received 2 September 2012
Received in revised form 6 December 2012
Accepted 14 December 2012
Available online 21 December 2012
Communicated by F.Y.L. Chin

Keywords:
Analysis of algorithms
Semi-online scheduling
Competitive ratio
Hierarchy

ABSTRACT

In this paper, we consider semi-online hierarchical scheduling problems on two identical machines, with the purpose of minimizing the makespan. The first investigated problem is the buffer version, where a buffer of a fixed capacity K is available for storing at most K jobs. When the current job is given, we are allowed to assign it on some machine irrecoverably; or temporarily store it in the buffer. But in the latter case if the buffer was full then an earlier job is removed from the buffer and assigned it to some machine. The second one is a reassignment version, where when the input is end, we are allowed to reassign at most K jobs. For both versions, we show no online algorithm can have a competitive ratio less than $\frac{3}{2}$, then propose two online algorithms with a competitive ratio $\frac{3}{2}$ with $K = 1$ for both versions of the problem, i.e., using only buffer of size one, or using only one rearrangement at the end.

1. Introduction

In this paper, we consider two versions of online hierarchical scheduling problems on two identical machines, with the purpose of minimizing the makespan. Jobs arrive one by one over a list L and each job has a positive size and hierarchy $g = 1$ or 2. There are two machines with the same speed to process these jobs. However, the machines have different capability, i.e., the first machine M_1 can process all the jobs while the second machine M_2 can process only jobs with $g_j = 2$.

Different from the full online version, we consider two semi-online versions. The first one is a buffer mode, where we can use a reordering buffer with a fixed capacity K in this model, and the buffer can store at most K jobs. When a new job arrives, we can put it in the buffer temporarily, or schedule it on one machine. In the first case, if the buffer is full with K jobs, then we have to kick out one job from the buffer and schedule the job on some machine.

The second one is a rearrangement version, where we are allowed to rearrange some jobs from M_2 to M_1, or from M_1 to M_2. After all the jobs have arrived and been scheduled, we are informed that the input sequence is over, then at most K already scheduled jobs can be reassigned.

The two versions (with buffer and with reassignment) of the problem seem closely related. But they are not equivalent with each other. Some discussions are given in Section 2.

We use *competitive ratio* to measure online or semi-online scheduling algorithms. For a job sequence J and a scheduling algorithm A, let $C^A(J)$ (C^A for short) be the makespane produced by A, and $C^*(J)$ (C^* for short) be the optimal makespan in an offline model. Then the *competitive ratio* of A is the infimum R such that for any input, $C^A \leqslant R \cdot C^*$. An online (or semi-online) version has a *lower bound* ρ if no online (or semi-online) scheduling algorithm has a competitive ratio smaller than ρ. An online (or semi-online) scheduling algorithm A is called *optimal* if

* Corresponding author.
E-mail addresses: cx.dlut@gmail.com (X. Chen), xzz@dlut.edu.cn (Z. Xu), dosagy@almos.vein.hu (G. Dósa), hanxin@dlut.edu.cn (X. Han), jianghe@dlut.edu.cn (H. Jiang).

0020-0190/$ – see front matter
http://dx.doi.org/10.1016/j.ipl.2012.12.007

its competitive ratio matches the lower bound of the version.

Related works Hierarchical scheduling on m parallel machines problem was first studied by Hwang et al. [6]. They studied the offline version and proposed an approximation algorithm LG-LPT with the makespan no more than $\frac{5}{4}$ times the optimum for $m = 2$, and $2 - \frac{1}{m-1}$ times the optimum for $m \geqslant 3$. For the online version, Park et al. [12] and Jiang et al. [8] independently presented optimal algorithms with a competitive ratio of $\frac{5}{3}$ for two identical machines. Jiang [7] further generalized the problem to m identical machines and proposed a lower bound and algorithms.

Several semi-online versions of this problem were studied these years. Park et al. [12] considered the version that the total processing time of all jobs is known in advance, and proposed an optimal algorithm with competitive ratio of $\frac{3}{2}$. Jiang et al. [8] considered the preemptive version and proposed an optimal solution too. Wu and Yang [16] investigated two different semi-online versions: the optimal offline value of the input is known in advance or the largest processing time of all jobs is known in advance. They proposed two algorithms with competitive ratios of $\frac{3}{2}$ and $\frac{1+\sqrt{5}}{2}$ for these two versions on two identical machines, which are shown to be optimal.

Using a buffer was widely investigated in scheduling problems [9,17,3,10]. Recently, a paper by Englert et al. [5] gave several results on identical or uniform machines. Since using a buffer is advantageous comparing to the situation when we cannot use it (for example, the best bound for scheduling two identical machines is 4/3 with a buffer and 3/2 without it, when the makespan is minimized [5]), we firstly consider the hierarchical scheduling problem with a buffer in this paper.

Secondly, we consider the rearrangement version, which was defined by Tan and Yu [13] and then investigated by several researchers [11,2,1,15]. There are two similar models: reassignment at the end and reassignment at any time [13]. For the reassignment at the end, Chen et al. proposed an optimal online algorithm for two related machines with $K = 2$ [2]. For the reassignment at any time, Dósa et al. obtained an optimal online algorithm for two related machines with $K = 2$ and an optimal online algorithm with $K = 1$ and $s \leqslant 1.3247$ or $s \geqslant 1.732$, where $s \geqslant 1$ is the speed of the faster machine [4].

Our results Consider the two versions we study in this paper. For the semi-online scheduling problem on two identical machines without hierarchy, the two versions have the same upper and lower bounds, respectively [2]. Whether the two versions also have the same tight bounds with hierarchy being under consideration, this is our motivation. For the both versions, we prove the lower bounds $\frac{3}{2}$, no matter what constant K is. Then we propose two optimal algorithms with $K = 1$ respectively, i.e., using only buffer of size one or using only one rearrangement at the end. Considering that the best solution of online version is $\frac{5}{3}$ [12,8], we can tell that having a buffer or rearrangement are both useful for hierarchical scheduling, and only buffer of size one or only one rearrangement is enough to achieve the best possible competitive ratio.

The rest of the paper is organized as follows. Section 2 gives some basic notations. Sections 3 and 4 propose lower bounds and optimal algorithms for the two versions, respectively. Some conclusion remarks are given in Section 5.

2. Preliminaries

In this paper, we are given a set of jobs $J = \{J_1, \ldots, J_n\}$ associated with positive sizes p_i and hierarchy $g_i = 1$ or 2, and two identical machines with different capability. The first machine M_1 can process all the jobs while the second machine M_2 can process only jobs with $g_i = 2$, which means that jobs with $g_i = 1$ must be scheduled on M_1. Then we should schedule J on M_1 and M_2 such that the maximal completion time of M_1 and M_2 is minimized.

The following notations and definitions are required in the remainder of the paper, for each $j = 1, 2, \ldots, n$ and $i = 1, 2$.

T_j: the total size of the first j jobs.
T_j^1: the total size of jobs with $g_i = 1$ in the first j jobs.
p_j^m: the largest job size among the first j jobs.
L_j^i: the total size of jobs scheduled on M_i after J_j is processed (scheduled on a machine or stored in the buffer).

Then we define $LB_j = \max\{p_j^m, T_j/2, T_j^1\}$ as the standard lower bound of the optimal makespan of the sequence containing the first j jobs. We also use (p_i, g_i) to denote job J_i, where p_i is the processing time and g_i is the hierarchy.

Lemma 1. *(See [8].) The optimal makespan is at least LB_j at any moment $j \geqslant 1$.*

Lemma 2. *The two models are not equivalent with each other.*

Proof. Let A_r be an algorithm which one can use one reassignment at the end, A_b be an online algorithm for a buffer of size 1. We first prove that there exists a list, for which by using a buffer of size $K = 1$ we can get an optimal solution, while using 1 rearrangement at the end we cannot.

The example is the next: the first job is $J_1 = (1, 1)$, where the first number denotes the size and the second number denotes the hierarchy. The job has to be assigned to M_1. We have no advantage if we put it into the buffer, thus both algorithms put it to M_1. Next $J_2 = (2, 2)$ is given. There are two cases.

Case 1. A_r assigns it to M_1. Then job $J_3 = (3, 2)$ is given. If J_3 is assigned on M_2, then $J_4 = (2, 2)$ is given. Else J_3 is assigned on M_1, then $J_4 = (4, 1)$ is given. Since A_r can do only one rearrangement, so it cannot reach an optimal solution (where the loads will equal). But A_b puts J_2 to M_2 when J_3 comes, and it can reach an optimal schedule in both subcases.

Case 2. A_r assigns J_2 to M_2. Then job $J_3 = (3, 2)$ is given. If J_3 goes to M_2, then job $J_4 = (6, 2)$ is given. Else J_3 goes