Contents lists available at ScienceDirect

European Journal of Operational Research

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

Discrete Optimization

Online scheduling of malleable parallel jobs with setup times on two identical machines ☆

Shouwei Guo, Liying Kang *

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

ARTICLE INFO

Article history:
Received 15 April 2009
Accepted 1 March 2010
Available online 6 March 2010

Keywords:
Scheduling
Parallel jobs
Setup times
Online algorithm
Competitive analysis

ABSTRACT

In this paper we consider online scheduling of malleable parallel jobs on two identical machines, where jobs arrive over time. Each job J_j has an execution time $t_j = p_j/k_j + (k_j - 1)c_j$ when it is processed on k_j machines, where $p_j > 0$ and $c_j > 0$ are the length and setup time of job J_j. The objective is to minimize the makespan. For the problem with two machines, we present an online algorithm with competitive ratio of $1 + \alpha$, where $\alpha = (\sqrt{5} - 1)/2$. We show that $1 + \alpha$ is a lower bound on the competitive ratio of any online algorithm for the problem with m $(m \geqslant 2)$ machines. So our algorithm is optimal for the case of two machines.

1. Introduction

In the considered online scheduling problem we are given a set of independent malleable parallel jobs and a set of identical machines. Jobs arrive over time. Let $J = \{J_1, J_2, \ldots, J_n\}$ be a set of n jobs. For $j = 1, 2, \ldots, n$, job J_j is characterized by three parameters, arrival time $a_j \geqslant 0$, length $p_j > 0$ and setup time $c_j > 0$, which are not known in advance. Every job becomes available at its arrival time and each machine can handle at most one job at a time. Since the jobs are malleable parallel, they may require more than one machine simultaneously for their processing and they can distribute their workload among any number of the available machines. After starting the processing of a job, preemption of its processing is not allowed. The goal is to minimize the *makespan*, i.e., the maximum completion time. According to the scheduling notation introduced by Graham et al. [6], this model is denoted by $Pm|$ *online*, *p-job*, a_j, c_j, *malleable*$|C_{\max}$.

For an online algorithm, the *competitive ratio* ρ is often used to measure its performance. Given an instance I, the makespan of a schedule produced by an online algorithm $\mathscr{A}$ is denoted by $C^{\mathscr{A}}(I)$, and the corresponding optimal makespan is denoted by $C^*(I)$. The competitive ratio ρ is the smallest number such that $C^{\mathscr{A}}(I) \leqslant \rho C^*(I)$ for any instance I. For simplicity, we use $C^{\mathscr{A}}$ and C^* instead of $C^{\mathscr{A}}(I)$ and $C^*(I)$ if there is no confusion. An online scheduling problem has a lower bound ρ' on its competitive ratio if there is no online algorithm with a competitive ratio smaller than ρ'. An online algorithm is called optimal if its competitive ratio meets the lower bound.

In scheduling, there are two basic online models: jobs arrive over time and jobs are presented one by one. In the first case, jobs are available at their arrival time and an algorithm may delay scheduling a job while future jobs arrive. In the second case, the next job will not come until the current job has been scheduled, so jobs must be scheduled immediately without knowing any future jobs.

For the online problem of scheduling non-malleable parallel jobs on m identical machines where jobs are presented one by one and the objective is to minimize the makespan, Johannes [11] presented a 12-competitive algorithm and a lower bound of 2.25. An improved online algorithm with competitive ratio 7 was provided by Ye and Zhang [21]. This ratio was improved by Ye et al. [20] and Hurink and Paulus [10]. They independently gave algorithms with a competitive ratio of 6.6623. The later paper gave also a 2.8 competitive algorithm for the special case with three machines. Chan et al. [2] proved that a Greedy algorithm achieved a competitive ratio 2 on the case of two machines. Hurink and Paulus [9] showed that 2 is a tight lower bound on the case of two machines, and they improved the lower bound of the case of m machines to 2.43 in the same paper. Meanwhile, this was improved to 2.457 by Kern and Paulus [12].

For the online problem of scheduling non-malleable parallel jobs on m identical machines to minimize the makespan where jobs arrive over time, Chen and Vestjens [3] proved a lower bound

☆ Research was partially supported by the National Nature Science Foundation of China (No. 10971131), Shanghai Leading Academic Discipline Project (No. S30104) and Innovation fund of Shanghai University (No. SHUCX091044).

* Corresponding author. Tel.: +86 21 66135652; fax: +86 21 66133292.
E-mail addresses: guoshouwei@shu.edu.cn (S. Guo), lykang@shu.edu.cn (L. Kang).

0377-2217/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2010.03.005

1.347 on the case without preemption, and Johannes [11] showed that 6/5 is a lower bound on the case where preemption is allowed, and he proved that a list scheduling algorithm has a competitive ratio of 2 for both cases. An online algorithm with competitive ratio $2 - 1/m$ was raised by Naroska and Schwiegelshohn [17] for the model where the processing times of jobs are not known until they are finished. This algorithm is optimal since Shmoys et al. [18] showed a lower bound $2 - 1/m$ on the competitive ratio of any online algorithm. More studies on the topic of parallel machines and parallel jobs can be found in [4,5,8,13–16,19].

In practice, it is often that some work should be done before we execute a job. So setup time is a considerable parameter in the scheduling problems [1,7]. The problem of online scheduling malleable parallel jobs with setup times was studied by Havill and Mao [7]. The ideal execution time of a malleable parallel job with length p is p/k if it utilizes k identical machines. However, delivering information from one machine to the others often prevent actual execution times from achieving this ideal. They considered setup time and define the execution time of job J_j processed on k_j machines to be $t_j = p_j/k_j + (k_j - 1)c$, where $c > 0$ is the setup time required to manage one more processor. At each time a job is assigned to k_j machines, the "master" processor should send a message to the other $k_j - 1$ "child" processors. Then p_j/k_j represents the parallel computation time and $(k_j - 1)c$ represents the time needed to complete message passing (setup). For the model of scheduling on m identical machines to minimize the makespan where jobs arrive over time, they presented an online algorithm called SET with competitive ratio $4(m-1)/m$ for even $m \geqslant 2$ and $4m/(m+1)$ for odd $m \geqslant 3$.

In practise, setup time is often related to not only the machine environment but also the job itself. Malleable parallel jobs may have different setup times to be delivered to one more processor. An exclusive setup time for each malleable parallel job is naturally to be considered. In this paper we study online scheduling of malleable parallel jobs with different setup times. The *execution time* of job J_j processed on k_j machines is defined as $t_j = p_j/k_j + (k_j - 1)c_j$, where c_j is an exclusive setup time of job J_j. Jobs arrive over time and preemption is not allowed. Our goal is to minimize the makespan. For the case of two identical machines, it is easy to see that $t_j = p_j$ when $k_j = 1$ and $t_j = p_j/2 + c_j$ when $k_j = 2$. We present an online algorithm with competitive ratio $1 + \alpha$ for the case of two identical machines, where $\alpha = (\sqrt{5} - 1)/2$ is the positive solution of equation $\alpha^2 + \alpha = 1$. Furthermore, we show that there is no online algorithm with competitive ratio smaller than $1 + \alpha$ on the case of m $(m \geqslant 2)$ identical machines. So our online algorithm is optimal for the case of two machines. Since setup times in our model may be different and an algorithm may be allowed to delay scheduling a job, the proof of our lower bound does not apply to the problem studied by Havill and Mao [7].

2. An on-line algorithm for two machines

In this section we consider the online scheduling of malleable parallel jobs on two identical machines. We present an online algorithm called STA (scheduling types alternatively) and show that the competitive ratio of the algorithm is no more than $1 + \alpha$. The following technical lemma is useful in the proof of the main theorem.

Lemma 1. *If* $0 \leqslant x' \leqslant x,\ 0 \leqslant y \leqslant y'$ *and* $\frac{a}{c} \leqslant \frac{b}{d}$, *where* $a, b, c, d \geqslant 0$, *then*

(i) $$\frac{ax + by}{cx + dy} \leqslant \frac{ax' + by'}{cx' + dy'}, \tag{2.1}$$

(ii) $$\frac{ax + by}{cx + dy} \leqslant \frac{b}{d}. \tag{2.2}$$

Proof

(i) $$\frac{ax+by}{cx+dy} = \frac{a}{c} + \frac{(b - \frac{ad}{c})y}{cx+dy} \leqslant \frac{a}{c} + \frac{(b - \frac{ad}{c})y}{cx'+dy} = \frac{ax'+by}{cx'+dy} = \frac{b}{d} + \frac{(a - \frac{bc}{d})x'}{cx'+dy} \leqslant \frac{b}{d} + \frac{(a - \frac{bc}{d})x'}{cx'+dy'} = \frac{ax'+by'}{cx'+dy'},$$

where the first inequality holds since $b \geqslant \frac{ad}{c},\ x \geqslant x'$, and the second inequality holds since $a \leqslant \frac{bc}{d},\ y \leqslant y'$.

(ii) By (i) and $x \geqslant 0$, we have

$$\frac{ax+by}{cx+dy} \leqslant \frac{by}{dy} = \frac{b}{d}. \qquad \square$$

We first identify two types of jobs to simplify the presentation of the algorithm. *Type I* jobs are considered as jobs with 'large' communication costs compared to the processing time (more precisely that $c_j > (\alpha - 1/2)p_j$) and the remaining jobs as *type II* jobs. The algorithm always executes type I jobs on one machine and type II jobs on two machines. Thus, every type I job J_j has execution time $t_j = p_j$ and every type II job J_j has execution time $t_j = p_j/2 + c_j \leqslant \alpha p_j$. In our algorithm, the basic idea is to have 'long blocks' of consecutive jobs of the same type and that type II jobs have preference over type I jobs if a new 'consecutive' block has to be started. If we start a 'block' with type II jobs we continue scheduling type II jobs as long as no idle times occur. While, if we schedule type I jobs, we schedule them in an arbitrary order and continue scheduling type I jobs as long as at least one machine is busy. The details of the algorithm are presented as follows.

For any given schedule σ, let

$U_1(t)$	set of unfinished type I jobs available at time t
$U_2(t)$	set of unfinished type II jobs available at time t

Algorithm STA

```
Step 0. Set $t = 0$.
Step 1. Determine $U_1(t)$ and $U_2(t)$.
Step 2. If $U_2(t) = \emptyset$, then go to Step 3; otherwise, schedule type II
        jobs one by one on two machines until the time $t'$ with
        $U_2(t') = \emptyset$. Set $t := t'$, back to Step 1.
Step 3. If $U_1(t) = \emptyset$, then go to Step 4; otherwise, schedule type I
        jobs in an arbitrary order until the time $t'$ with
        $U_1(t') = \emptyset$. Set $t := t'$, back to Step 1.
Step 4. If there are still some jobs to arrive, set $t$ as the arrival time
        of the next job and back to Step 1; otherwise, stop and
        complete the schedule at time $t$.
```

For the schedule produced by Algorithm STA, a *part B* is defined as a maximal time interval in which there is no period of idle time on both machines. A *type I block* is a maximal time interval that contains only type I jobs in a part and a *type II block* is a maximal time interval that contains only type II jobs in a part. We can get the following observations.

Observation 1. *The schedule produced by Algorithm STA can be naturally partitioned into N parts. Let $B_1, \ldots, B_N$ be these parts. Every part B_i consists of alternating blocks of different types: type I blocks and type II blocks. The blocks follow directly after each other and within type I blocks idle times may occur but not on both machines simultaneously (see Fig. 1).*

Observation 2. *All jobs in a type I block (type II block) must arrive at or after the beginning time of the type II block (type I block) immediately scheduled before it, if there is one.*