Information Processing Letters 114 (2014) 179–184

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

Online scheduling on an unbounded parallel-batch machine and a standard machine to minimize makespan

Ruyan Fu [a], Ji Tian [a], Jinjiang Yuan [b,*], Ya Li [b]

[a] College of Sciences, China University of Mining and Technology, Xuzhou, Jiangsu 221116, People's Republic of China
[b] School of Mathematics and Statistics, Zhengzhou University, Zhengzhou, Henan 450001, People's Republic of China

ARTICLE INFO

Article history:
Received 15 October 2012
Received in revised form 24 November 2013
Accepted 24 November 2013
Available online 26 November 2013
Communicated by B. Doerr

Keywords:
Scheduling
Online algorithm
Makespan
Parallel batch machine

ABSTRACT

We consider the online scheduling on an unbounded parallel-batch machine and a standard machine to minimize makespan. In the problem, the jobs arrive online over time and to be processed on two machines M_1 and M_2. M_1 is an unbounded parallel-batch machine and M_2 is a standard machine. The objective is to minimize the makespan, i.e., the maximum completion time of all jobs. For this problem, we present an online algorithm of competitive ratio $\frac{5+\sqrt{5}}{4} \approx 1.809$.

1. Introduction

In the scheduling on parallel-batch machines, the jobs can be processed in batches with capacity b, where a batch is a subset of at most b jobs. The processing time of a batch is defined to be the maximum processing time of the jobs in the batch. When $b = 1$, the parallel-batch machine degenerates as a standard machine.

We consider the online scheduling on two machines M_1 and M_2, where M_1 is an unbounded parallel-batch machine (i.e., $b = \infty$) and M_2 is a standard machine. In our research, the online version means that the jobs arrive over time and the jobs' information is unknown until their arrival times. The processing time and the arrival time of a job J_j are denoted by p_j and r_j, respectively, where $p_j > 0$ and $r_j \geqslant 0$. The released and not yet started jobs can be processed either on the batch machine M_1 or on the standard machine M_2. Preemption and restart are not allowed in the problem. That is, once the processing of a batch or a job begins, the processing cannot be interrupted until completed. The objective is to minimize the makespan, i.e., the maximum completion time of all jobs. Throughout this paper, we use $\mathcal{P}(1, \infty)$ to denote the problem in consideration.

The research in this paper is motivated by the logistics problem in online shopping. An express company has several means of transportation that have different loading capacities. Each order (task) from a customer is of the form to deliver its goods to the goal place. The delivery time of an order is the time used in a round of transportation of the order which includes the time used in the mean to deliver the goods in the order to the goal place and to return to the company. Assume that the means have equal speed in their transportation, each order occupies a unit capacity of the mean to deliver it. When multiple orders are delivered by a common mean in a batch, the delivery time of this batch is equal to the maximum delivery time of the orders in this batch. In practice, the orders from the customers usually arrive online at any time and the company expects to complete the transportation of all orders in the earliest time. By regarding means of transportation as batch machines, the orders as jobs, the loading capacity of a mean as the batch capacity, and the delivery times of the orders as their processing times, we back to a

* Corresponding author.
E-mail address: yuanjj@zzu.edu.cn (J. Yuan).

0020-0190/$ – see front matter
http://dx.doi.org/10.1016/j.ipl.2013.11.016

scheduling problem on multiple batch machines with distinct capacities to minimize the makespan. In this paper, we study the basic version in which there are two batch machines with the capacities 1 and ∞, respectively.

The quality of an online algorithm is measured by its competitive ratio. An online algorithm is called ρ-competitive if the objective value of the schedule given by the online algorithm is at most ρ times of that of the optimal off-line schedule. An online algorithm is called best possible if its competitive ratio matches the lower bound of the competitive ratio of the problem.

Scheduling on identical machines and parallel-batch machines has been studied extensively. Representative publications can be found in Chen and Vestjens [2], Noga and Seiden [6], Deng et al. [3], Poon and Zhang [7], Poon and Yu [8]. For online scheduling on identical unbounded parallel-batch machines to minimize makespan, Zhang et al. [11] provided a lower bound of competitive ratio of $\sqrt[m+1]{2}$, and presented an online algorithm with a competitive ratio of $1+\beta_m$, where $0<\beta_m<1$ is the positive solution of equation $\beta_m=(1-\beta_m)^{m-1}$. In the special case where all the jobs have equal processing times, Zhang et al. [12] presented two best possible online algorithms with competitive ratios of $1+\xi_m$ and $\frac{\sqrt{5}+1}{2}$ for unbounded version and bounded version, respectively, where ξ_m is the positive solution of equation $(1+\xi_m)^{m+1}=\xi_m+2$. For the general version, Liu et al. [5] and Tian et al. [9] using different approaches provided online algorithms with the best possible competitive ratio of $1+\alpha_m$, where α_m is the positive root of $\alpha^2+m\alpha-1=0$.

As far as we know, the first famous paper of scheduling on batch and discrete processors is Ahmadi et al. [1]. They considered a class of scheduling problems arising from a two-machine flow shop system, where each system has at least one batch machine. The objective is to minimize the makespan and the sum of job completion times. The authors studied the two-machine system that has a standard machine, followed by a batch machine. They analyzed the complexity of the problems, proposed polynomial-time procedures for some special cases and provided heuristics for the NP-hard versions.

The online scheduling problem $\mathcal{P}(1,\infty)$ was first studied by Yao [10]. For the problem, Yao [10] showed that no online algorithm has a competitive ratio of less than $(\sqrt{5}+1)/2\approx 1.618$, and provided an online algorithm of competitive ratio 2. When the jobs satisfy the condition "$r_i<r_j\Rightarrow p_i\leqslant p_j$", Li [4] presented a best possible online algorithm of competitive ratio $(\sqrt{5}+1)/2$.

The main result of our paper is to improve the upper bound of competitive ratio of problem $\mathcal{P}(1,\infty)$. We provide an online algorithm of competitive ratio $\frac{5+\sqrt{5}}{4}\approx 1.809$. This leaves a gap from 1.618 to 1.809.

2. The online algorithm

For online algorithms of problem $\mathcal{P}(1,\infty)$, the following lower bound $(\sqrt{5}+1)/2$ of the competitive ratios was first established in Yao [10]. We present a short proof for completion.

Denoted by $C(\sigma)$ and $C(\pi)$ the objective values given by an online algorithm and an optimal off-line algorithm, respectively. Set $\alpha=\frac{\sqrt{5}-1}{2}$.

Theorem 2.1. *(See Yao [10].) For problem $\mathcal{P}(1,\infty)$, no online algorithm has a competitive ratio less than $1+\alpha=(\sqrt{5}+1)/2$. The result still holds even when all jobs have an identical processing time.*

Proof. Assume that each job has a processing time 1. Suppose that two jobs J_1 and J_2 arrive at time 0. If the online algorithm starts no jobs before time α on the batch machine M_1, then no new jobs will arrive. We have $C(\sigma)\geqslant 1+\alpha$ and $C(\pi)=1$, and so, $C(\sigma)/C(\pi)\geqslant 1+\alpha$.

Suppose that the online algorithm starts at least one of J_1 and J_2 at a time $t<\alpha$ on M_1. Then two new jobs arrive at time $t+\epsilon$, where ϵ is a sufficiently small positive number. Assume that no other jobs will arrive. Then $C(\sigma)\geqslant t+2$ and $C(\pi)=t+\epsilon+1$, and so, $C(\sigma)/C(\pi)\geqslant 1+(1-\epsilon)/(t+\epsilon+1)>1+(1-\epsilon)/(\alpha+\epsilon+1)\to 1+\alpha$ as $\epsilon\to 0$. The result follows. □

For a set of jobs V, a job $J_i\in V$ is called *the last longest job* in V if $p_i=\max\{p_j: J_j\in V\}$ so that r_i is as large as possible. For a time instant t and an online algorithm, we use $U(t)$ to denote the set of released and not yet started jobs at time t. Let $J'(t)$ be the last longest job in $U(t)$ and $J''(t)$ the last longest job in $U(t)\setminus\{J'(t)\}$. The processing times and arrival times of $J'(t)$ and $J''(t)$ are denoted by $p'(t)$, $p''(t)$, $r'(t)$ and $r''(t)$, respectively. We assume $p'(t)>0$ if $U(t)\neq\emptyset$. In the case $U(t)=\{J'(t)\}$, we define $J''(t)$ to be a dummy job with $p''(t)=0$ and $r''(t)=r'(t)$. A batch processed on M_1 or a job processed on M_2 is called a task in the online algorithm. If just one machine is busy at time t, we use $p^*(t)$ to denote the processing time of the running task, say $T^*(t)$, at time t. The starting time of $T^*(t)$ is denoted by $S^*(t)$. If both machines are idle at time t, we define $p^*(t)=0$ and $S^*(t)=0$.

Set $\beta=\frac{\alpha+1}{2}$. It can be observed that $\frac{1}{\beta}=2\alpha$ and $\alpha^2+\alpha=1$.

Our online algorithm can be described as shown in Algorithm H.

Let t be the starting time of a task in Algorithm H. From the implementation of H, either a batch B starting at time t on M_1 or a job J starting at time t on M_2. Let $x,y\in\{0,1\}$ with $(x,y)\neq(0,0)$. We say that t is an (x,y)-type starting time in H if x batches starting at time t on M_1 and y jobs starting at time t on M_2 simultaneously in H. Clearly, every starting time in H is either of $(1,1)$-type, or of $(1,0)$-type, or of $(0,1)$-type. Furthermore, from Step 2 and Step 3 of Algorithm H, if a starting time t in H is of $(1,0)$-type or $(1,1)$-type, then all jobs in $U(t)$ will be started at time t in H. The following observations are implied in the implementation of Algorithm H, which may help the readers follow the line.

Observation 1. *Let t be a $(1,1)$-type starting time in Algorithm H. Then we have*