Intelligent Information Management, 2016, 8, 98-102
Published Online July 2016 in SciRes. http://www.scirp.org/journal/iim
http://dx.doi.org/10.4236/iim.2016.84008

On-Line Scheduling for Jobs with Arbitrary Release Times on Parallel Related Uniform Machines

Xiayan Cheng[1], Rongheng Li[2*], Yunxia Zhou[3]

[1]Department of Mathematics, Hunan First Normal University, Changsha, China
[2]Department of Mathematics, Key Laboratory of High Performance Computing and Stochastic Information Processing, Hunan Normal University, Changsha, China
[3]Department of Computer, Hunan Normal University, Changsha, China
Email: *lirongheng@hunnu.edu.cn

Received 18 February 2016; accepted 22 July 2016; published 25 July 2016

Abstract

A parallel related uniform machine system consists of *m* machines with different processing speeds. The speed of any machine is independent on jobs. In this paper, we consider online scheduling for jobs with arbitrary release times on the parallel uniform machine system. The jobs appear over list in terms of order. An order includes the processing size and releasing time of a job. For this model, an algorithm with competitive ratio of 12 is addressed in this paper.

Keywords

Online Scheduling, Uniform Machine, Competitive Ratio, Approximation Algorithm

1. Introduction

For the online scheduling on a system of *m* uniform parallel machines, denoted by $Q_m/online/C_{\max}$, each machine $M_i\left(i=1,2,\cdots,m\right)$ has a speed s_i, *i.e.*, the time used for finishing a job with size *p* on M_i is p/s_i. Without loss of generality, we assume $s_1 \le s_2 \le \cdots \le s_m$. Cho and Sahni [1] are the first to consider the on-line scheduling problem on *m* uniform machines. They showed that the LS algorithm for $Q_m/online/C_{\max}\left(m\ge 2\right)$ has competitive ratio not greater than $1+\sqrt{m-1}\big/\sqrt{2}$. When $s_i=1\left(i=1,2,\cdots,m-1\right)$ and $s_m=s>1$, they proved that the algorithm LS has a competitive ratio $1+\dfrac{m-1}{m+s-1}\min\left\{s,2\right\}\le 3-\dfrac{4}{m+1}$ and the bound

*Corresponding author.

How to cite this paper: Cheng, X.Y., Li, R.H. and Zhou, Y.X. (2016) On-Line Scheduling for Jobs with Arbitrary Release Times on Parallel Related Uniform Machines. *Intelligent Information Management*, **8**, 98-102.
http://dx.doi.org/10.4236/iim.2016.84008

$3-\dfrac{4}{m+1}$ is achieved when $s=2(m\ge 3)$.

For $Q_2/online/C_{\max}$, Epstein *et al.* [2] showed that LS has the competitive ratio $\min\left\{\dfrac{2s+1}{s+1},\dfrac{s+1}{s}\right\}$ and is an optimal online algorithm, where the speed ratio $s=s_2/s_1$.

Cai and Yang [3] considered $Q_3/online/C_{\max}$. Let $s=s_2/s_1$ and $t=s_3/s_2$ be two speed ratios. They showed that the algorithm *LS* is an optimal online algorithm when the speed ratios $(s,t)\in G_1\cup G_2$, where $G_1=\left\{(s,t)\,|\,1\le t<\dfrac{1+\sqrt{31}}{6},s\ge\dfrac{3t}{5+2t-6t^2}\right\}$ and $G_2=\{(s,t)\,|\,t\ge 1+s,s\ge 1,t\ge 1\}$. Moreover, for the general speed ratios, they also presented an upper bound of the competitive ratio.

Aspnes *et al*. [4] are the first to try to design algorithms better than LS for $Q_m/online/C_{\max}$. They presented a new algorithm that achieves the competitive ratio of 8 for the deterministic version, and 5.436 for its randomized variant. Later the previous competitive ratios are improved to 5.828 and 4.311, respectively, by Berman *et al*. [5].

Li and Shi [6] proved that for $m\le 3$ LS is optimal when $s_i=1(i=1,2,\cdots,m-1)$ and $s_m=2$ and presented an online algorithm with a better competitive ratio than LS for $m\ge 4$. Besides, they also showed that the bound $3-\dfrac{4}{m+1}$ could be improved when $s_i=1(i=1,2,\cdots,m-1)$ and $s_m=s>1$. For $m\ge 4$ and $1\le s\le 2$, Cheng *et al*. [7] proposed an algorithm with a competitive ratio not greater than 2.45.

A generalization of the Graham's classical on-line scheduling problem on *m* identical machines was proposed by Li and Huang [8]-[10]. They describe the requests of all jobs in terms of order. For an order of the job J_j, the scheduler is informed of a 2-tuple (r_j,p_j), where r_j and p_j represent the release time and the processing time of the job J_j, respectively. The orders of request have no release time but appear on-line one by one at the very beginning time of the system. Whenever the request of an order is made, the scheduler has to assign a machine and a processing slot for it irrevocably without knowledge of any information of future job orders. In this on-line situation, the jobs' release times are assumed to be arbitrary.

Our task is to allocate a sequence of jobs to the machines in an on-line fashion, while minimizing the maximum completion time of the machines. In the following of this paper, *m* parallel uniform machines which have speeds of $s_1\le s_2\le\cdots\le s_m$, respectively, are given. Let $L=\{J_1,J_2,\cdots,J_n\}$ be any list of jobs, where job J_j is given as order with the information of a release time r_j and a processing size of p_j.

The rest of the paper is organized as follows. In Section 2, some definitions are given. In Section 3, an algorithm U is addressed and its competitive ratio is analyzed.

2. Some Definitions

In this section we will give some definitions.

Definition 1. *We have m parallel machines with speeds* $s_1,s_2,\cdots,s_m$. *Let* $L=\{J_1,J_2,\cdots,J_n\}$ *be any list of jobs, where jobs arrives online one by one and each* J_j *has a release time* r_j *and a processing size of* p_j. *Algorithm A is a heuristic algorithm. Let* $C_{\max}^{A}(L)$ *and* $C_{\max}^{OPT}(L)$ *be the makespan of algorithm A and the makespan of an optimal off-line algorithm respectively. We refer to*

$$R(m,A)=\sup_{L}\frac{C_{\max}^{A}(L)}{C_{\max}^{OPT}(L)}$$

as the competitive ratio of algorithm *A*.

Definition 2. *Suppose that* J_j *is the current job to be scheduled with release time* r_j *and size* p_j. *We say that machine* M_i *has an idle time interval for job* J_j, *if there exists a time interval* $[T_1,T_2]$ *satisfying the following two conditions*:

1) Machine M_i is idle in interval $[T_1,T_2]$ and a job with release time T_2 is assigned on machine M_i to start at time T_2.

2) $T_2-\max\{T_1,r_j\}\ge\dfrac{p_j}{s_i}$.

It is obvious that if machine M_i has an idle time interval for job J_j, then we can assign J_j to machine M_i in the idle interval.

In the following we consider m parallel uniform machines with speeds $s_1, s_2, \cdots, s_m$ and a job list $L = \{J_1, J_2, \cdots, J_n\}$ with information (r_j, p_j) for each job $J_j \in L$, where r_i and p_i represent its release time and size, respectively. For convenience, we assume that the sequence of machine speeds is non-decreasing, *i.e.*, $s_1 \le s_2 \le \cdots \le s_m$.

3. Algorithm U and Its Performance

Now we present the algorithm *U* by use of the notations given in the former section in the following:

Algorithm U:

Step 0. (*start the first phase*)

$$h := 1,\ j := 1,\ \vec{l} := \{0, 0, \cdots, 0\},\ \Delta_h := r_j + p_j / s_m .$$

Step 1. If there is a new job J_j with release time r_j and processing size p_j given to the algorithm then go to Step 2. Otherwise stop.

Step 2. If there is a machine M_i which has an idle time interval for job J_j, then we assign J_j to machine M_i in the idle interval. Set $j := j+1$ and go to Step 1.

Step 3. Set $S := \{i \mid \max\{l_i, r_j\} + p_j / s_i \le 3\Delta_h\}$. If $S \ne \varnothing$ then set $k := \min\{i \mid i \in S\}$, $l_k := \max\{l_k, r_j\} + p_k$, $l_i := l_i, i \ne k, j := j+1$. Go to Step 1.

Step 4. (*start a new phase*)

Set $\vec{l} = \{0, 0, \cdots, 0\}$, $\Delta := 2\Delta_h$. $h := h+1, \vec{l} := \{0, 0, \cdots, 0\}$, $\Delta_h := \Delta, j := j$ and go to Step 3.

Now we begin to analyze the performance of algorithm U.

The following statement is obvious:

Lemma 1. Let L_h be the stream of jobs scheduled in phase h and J_j is the first job assigned in phase $h+1$. Let Γ_h^* be the largest load in an optimal schedule for job list $L_h \cup \{J_j\}$. Then we have $\Gamma_h^* > \Delta_h$.

Proof: If $\Gamma_h^* \le \Delta$, let r be the fastest machine whose load does not exceed $2\Gamma_h^*$, *i.e.* $r = \max\{i \mid l_i(j-1) \le 2\Gamma_h^*\}$. If there is no such machine, we set $r = 0$. If $r = m$, then $l_m(j-1) \le 2\Gamma_h^*$. It is obvious that $r_j + \frac{p_j}{s_m} \le \Gamma_h^*$. Hence we have

$$\max\{r_j, l_m(j-1)\} + \frac{p_j}{s_m} \le 2\Gamma_h^* + \Gamma_h^* \le 3\Delta_h.$$

It means that J_j can be assigned to the fastest machine M_m in phase *h*. It is a contradiction to the fact that J_j is the first job assigned in phase $h+1$. Define $\beta = \{i \mid i > r\}$, the set of machines with finishing time bigger than $2\Gamma_h^*$ by the end of phase *h*. Since $r < m$, $\beta \ne \varnothing$. Denote by S_i and S_i^* the set of jobs assigned to machine M_i by the on-line and the off-line algorithms, respectively. Since for any job $J_u \in L_h$ the following inequalities hold

$$r_u < r_u + \frac{p_u}{s_m} \le \Gamma_h^*,$$

we get:

$$\frac{1}{s_i} \sum_{u \in S_i} p_u > \Gamma_h^*, \qquad \forall i \in \beta.$$

That means:

$$\frac{\sum_{i \in \beta} \sum_{u \in S_i} p_u}{\sum_{i \in \beta} s_i} > \Gamma_h^*.$$

This implies that there exists a job J_u ($u \in \bigcup_{i \in \beta} S_i$) such that $u \notin \bigcup_{i \in \beta} S_i^*$, *i.e.* there exists a job J_u assigned by the on-line algorithm to a machine $M_i(i \in \beta)$ and assigned by the off-line algorithm to a slower machine $M_{i'}\left(i' \notin \beta\right)$.

By our assumptions, we have $r_u + \frac{p_u}{s_{i'}} \le \Gamma_h^* \le \Delta_h$. Since $r \ge i'$, machine M_r is at least as fast as machine $M_{i'}$, and thus $r_u + \frac{p_u}{s_r} \le \Gamma_h^* \le \Delta_h$. Since job J_u was assigned before job J_j and $i' \notin \beta$, we have

$$l_r\left(u-1\right) \le l_r\left(j-1\right) \le 2\Gamma_h^*.$$

This implies

$$\max\left\{r_u, l_r\left(u-1\right)\right\} + \frac{p_u}{s_r} \le \max\left\{r_u, l_r\left(j-1\right)\right\} + \frac{p_u}{s_{i'}} \le l_r\left(j-1\right) + r_u + \frac{p_u}{s_{i'}} \le 3\Gamma_h^*.$$

But this means that the on-line algorithm should have placed job J_u on M_r or a slower machine instead of M_i, which is a contradiction. ■

Theorem 2. Algorithm achieves a competitive ratio of 12.

Proof: Let ρ_h denote the maximum load generated by jobs that were assigned during phase *h*; denote the last phase by h_{last}. By the rules of our algorithm we have $\Delta_h = 2^{h-1}\Delta_1$ and

$$\rho_h < 3\Delta_h = 3\cdot 2^{h-1}\Delta_1.$$

Hence the total height generated by the assignment is:

$$\Gamma = \sum_{h=1}^{h_{last}} \rho_h \le 3\left(2^{h_{last}} - 1\right)\Delta_1.$$

The claim of the theorem is trivially true if $h_{last} = 1$. For $h > 1$, phase *h* is started only if $\Gamma_{h-1}^* > \Delta_{h-1}$. In particular we have

$$\Gamma^* \ge \Gamma_{h_{last}-1}^* > \Delta_{h_{last}-1} = \Delta_{h_{last}}/2 = 2^{h_{last}-2}\Delta_1.$$

Therefore we have

$$\Gamma \le 3\left(2^{h_{last}} - 1\right)\Delta_1 < 12\Gamma^*.$$ ■

4. Concluding Remarks

In this paper, we consider on-line scheduling for jobs with arbitrary release times on uniform machines. An algorithm with the competitive ratio of 12 is given. It should be pointed out that more detailed consideration should be taken in order to improve the competitive ratio.

Acknowledgements

The authors would like to express their thanks to the National Natural Science Foundation of China for financially supporting under Grant No. 11471110 and No. 61271264.

References

[1] Cho, Y. and Sahni, S. (1980) Bounds for List Schedules on Uniform Processors. *SIAM Journal on Computing*, **9**, 91-103. http://dx.doi.org/10.1137/0209007

[2] Epstein, L., Noga, J., Seiden, S.S., Sgall, J. and Woeginger, G.J. (2001) Randomized On-Line Scheduling on Two Uniform Machines. *Journal of Scheduling*, **4**, 71-92. http://dx.doi.org/10.1002/jos.60

[3] Cai, S.Y. and Yang, Q.F. (2012) Online Scheduling on Three Uniform Machines. *Discrete Applied Mathematics*, **160**, 291-302. http://dx.doi.org/10.1016/j.dam.2011.10.001

[4] Aspnes, J., Azar, Y., Fiat, A., Plotkin, S. and Waarts, O. (1997) On-Line Routing of Virtual Circuits with Applications to Load Balancing and Machine Scheduling. *Journal of the ACM*, **44**, 486-504. http://dx.doi.org/10.1145/258128.258201

[5] Berman, P., Charikar, M. and Karpinski, M. (2000) On-Line Load Balancing for Related Machines. *Journal of Algo-*

rithms, **35**, 108-121. http://dx.doi.org/10.1006/jagm.1999.1070

[6] Li, R.H. and Shi, L.J. (1998) An On-Line Algorithm for Some Uniform Processor Scheduling. *SIAM Journal on Computing*, **27**, 414-422. http://dx.doi.org/10.1137/S0097539799527969

[7] Cheng, T.C.E., Ng, C.T. and Kotov, V. (2006) A New Algorithm for Online Uniform-Machine Scheduling to Minimize the Makespan. *Information Processing Letters*, **99**, 102-105. http://dx.doi.org/10.1016/j.ipl.2006.02.012

[8] Li, R.H., Cheng, X.Y. and Zhou, Y.X. (2014) On-Line Scheduling for Jobs with Non-Decreasing Release Times and Similar Lengths on Parallel Machines. *Optimization*: *A Journal of Mathematical Programming and Operations Research*, **63**, 867-882. http://dx.doi.org/10.1080/02331934.2014.895902

[9] Li, R.H. and Huang, H.C. (2004) On-Line Scheduling for Jobs with Arbitrary Release Times. *Computing*, **73**, 79-97. http://dx.doi.org/10.1007/s00607-004-0067-1

[10] Li, R.H. and Huang, H.C. (2007) Improved Algorithm for a Generalized On-Line Scheduling Problem on Identical Machines. *European Journal of Operations Research*, **176**, 643-652. http://dx.doi.org/10.1016/j.ejor.2005.06.061