International Journal of Intelligence Science, 2019, 9, 59-65
http://www.scirp.org/journal/ijis
ISSN Online: 2163-0356
ISSN Print: 2163-0283

Better Algorithm for Order On-Line Scheduling on Uniform Machines

Rongheng Li[1], Yunxia Zhou[2]

[1]Key Laboratory of Computing and Stochastic Mathematics (Ministry of Education), School of Mathematics and Statistics, Department of Mathematics, Hunan Normal University, Changsha, China
[2]Department of Computer, Hunan Normal University, Changsha, China
Email: lirongheng@hunnu.edu.cn

How to cite this paper: Li, R.H. and Zhou, Y.X. (2019) Better Algorithm for Order On-Line Scheduling on Uniform Machines. *International Journal of Intelligence Science*, **9**, 59-65.
https://doi.org/10.4236/ijis.2019.92004

Received: March 5, 2019
Accepted: April 27, 2019
Published: April 30, 2019

Abstract

In this paper, we consider online scheduling for jobs with arbitrary release times on the parallel uniform machine system. An algorithm with competitive ratio of 7.4641 is addressed, which is better than the best existing result of 12.

Keywords

Online Scheduling, Uniform Machine, Competitive Ratio, LS 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 $s_1 < s_2 \le \cdots \le s_m$. Cho and Sahni [1] are the first to consider the online scheduling problem on m uniform machines. For Q_2/online/$C_{\max}$, Epstein *et al.* [2] showed that *LS* has the competitive ratio $\min\left\{(2s+1)/(s+1),(s+1)/s\right\}$ and is an optimal online algorithm, where the speed ratio $s=s_2/s_1$. Q_3/online/$C_{\max}$ was considered by Cai and Yang [3]. They showed that the algorithm *LS* is an optimal online algorithm when the speed ratios $(s,t)\in G_1\cup G_2$, where $s=s_2/s_1$, $t=s_3/s_2$,

$$G_1=\left\{(s,t)\middle|1\le t<\frac{1+\sqrt{31}}{6},s\ge\frac{3t}{5+2t-6t^2}\right\},$$

$$G_2=\left\{(s,t)\middle|t\ge 1+s,s\ge 1,t\ge 1\right\}.$$

Aspnes *et al.* [4] are the first to try to design better algorithm for Q_m/online/$C_{\max}$.

They presented a new algorithm with competitive ratio of 8 for the deterministic version, and 5.436 for its randomized variant. Later the previous ratios are improved to 5.828 and 4.311, respectively, by Berman *et al.* [5].

The special case $s_i = 1(i = 1, 2, \cdots, m-1)$ and $s_m = s \geq 1$ was fisrt considered by Li and Shi [6]. It is proved that for $m \leq 3$ LS is optimal when $s_m = 2$ and they also developed an algorithm with a better competitive ratio than LS for $m \geq 4$ and $s_m = s \geq 1$. For $m \geq 4$ and $1 \leq s \leq 2$, Cheng *et al.* [7] proposed an algorithm with a competitive ratio not greater than 2.45.

Motivated by air cargo import terminal problem, a generalization of the Graham's classical on-line scheduling problem was proposed by Li and Huang [8] [9]. They describe the requests of all jobs in terms of order, where for any job list $L = \{J_1, J_2, \cdots, J_n\}$, job J_j is given as order with the information of a release time r_j and a processing size of p_j. More recent results can be found in the research by Li *et al.* [10] and Yin *et al.* [11].

Our task is to allocate an order sequence of jobs to m parallel uniform machines which have speeds of $s_1 \leq s_2 \leq \cdots \leq s_m$ in an online fashion, while minimizing the maximum completion time of the machines. An algorithm with worst case performance not bigger than 7.4641 is developed. The result is better than the existing result of 12 in Cheng *et al.* [12].

The rest of the paper is organized as follows. In Section 2, some definitions are given. In Section 3, an algorithm R 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 $s_1, s_2, \cdots, s_m$, respectively. 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. $C_{\max}^{A}(L)$ and $C_{\max}^{OPT}(L)$ denote the makespan of algorithm A and 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 with release time r_j and size of 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 has been assigned to machine M_i to start at time T_2.

2) $T_2 - \max[T_1, r_j] \geq \frac{p_j}{s_i}$.

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 processing size, respectively. For convenience, we assume that the sequence of machine speeds is non-decreasing, *i.e.*, $s_1 \le s_2 \le \cdots \le s_m$. Let

$$S = \{0, s_1, s_2, \cdots, s_m\};$$

$$\text{Bigsum}(s) = \sum_{s_i \in S, s_i > s} s_i;$$

$$S_i = \{0, s_i, s_{i+1}, \cdots, s_m\}; \quad i = 1, 2, \cdots, m, m+1;$$

$$\text{Bigsum}(i, s) = \sum_{s_{i'} \in S, s_{i'} > s} s_{i'}; \quad i = 1, 2, \cdots, m,$$

$$\text{Bigsum}(s, \Delta, L) = \sum_{J_j \in L, sr_j + p_j > s\Delta} p_j.$$

By our definition, $S_{m+1} = \{0\}$.

3. Algorithm R and Its Performance

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

Algorithm R:

```
Step 0.  Let $t$: = 0, $\Delta_t$: = very small positive number.
         For $i$ = 1 to $m$ do $m_i$: = $\Delta_t s_i$, $c_i$: = 2 $m_i$, $H_i$ = 0.
         $L^t := \Phi$.
Step 1.  Let $J_j$ be a new job with release time $r_j$ and processing size $p_j$ given
         to the algorithm. 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 in-
         terval and set $L^t := L^t \cup J_j$.
Step 2.  If $r_j \ge \Delta_t$ or $\text{Bigsum}\left(s, \Delta_t, \bigcup_{k=0}^{t} L^k\right) > \Delta_t \text{Bigsum}(s)$ for some
         $s \in S$ then goto Step 3. Otherwise goto Step 4.
Step 3.  (*start a new phase*)
         Set $\Delta$: = $r\Delta_t$, $t$: = $t$ + 1, $\Delta_t$: = $\Delta$,
         For $i$ = 1 to $m$ do $m_i := \Delta_t s_i$, $c_i := c_i + (2 - 1/r) m_i$.
         Set $L^t := \Phi$ and Goto Step 2.
Step 4.  (*schedule $p_j$*)
         $k := \min\{i \mid c_i + m_i > s_i r_j + p_j\}$;
         Assign $J_j$ on machine $M_k$; Set:
         $c_k := c_k - \max\{0, s_k r_j - H_k\} - p_j$;
         $H_k := H_k + \max\{0, s_k r_j - H_k\} + p_j$.
         Set $L^t := L^t \cup J_j$.
```

The running time of R is mainly in Step 1 and Step 4. In Step 1, at most j-1 times of checking can determine if there is an idle interval for current job J_j. In Step 4 at most m times can determine to assign current job J_j. Hence the complexity of our algorithm is O(n^2).

Now we begin to analyze the performance of algorithm R. The following statement is obvious:

Lemma 1. For a job list *L*, if $C_{\max}^{OPT}(L) \le \Delta$, then $\text{Bigsum}(s,\Delta,L) \le \Delta\text{Bigsum}(s)$ and $r_j < \Delta$ hold for every $s \in S$ and every $j \in L$.

Let L^l be the stream of jobs scheduled in phase *l*. We define L_i^l to be the stream of jobs that in phase *l* machine M_i passed over or M_{i+1} received, $i = 0,1,\cdots,m$ (for $1 \le i < m$, these two conditions are equivalent, for *i* = 0, only the latter and for *i* = *m* only the former applies).

Now the correctness of algorithm R will mean that the stream J_m^l is empty for every phase *l*.

Lemma 2. For every $i = 0,1,2,\cdots,m$ and every phase l, we have

$$\sum_{t=0}^{l} \text{Bigsum}\left(0,\Delta_t,L_i^t\right) \le \left(\sum_{t=0}^{l}\Delta_t\right)\left(\sum_{q=i+1}^{n} s_q\right).$$

Proof: First of all, by the rules of the algorithm, we have

$$\text{Bigsum}\left(s,\Delta_t,\bigcup_{t=0}^{l} L^t\right) \le \Delta_l\text{Bigsum}(s).$$

for every phase *l* and every $s \in S$. Therefore

$$\text{Bigsum}\left(s,\Delta_t,L^l\right) \le \Delta_l\text{Bigsum}(s).$$

for every phase *l* and every $s \in S$. Now we prove the claim by induction. For *i* = 0, it follows simply from the fact that

$$\text{Bigsum}\left(0,\Delta_t,L_0^t\right) = \text{Bigsum}\left(0,\Delta_t,L^t\right) \le \Delta_t\text{Bigsum}(0) = \Delta_t\sum_{q=1}^{m} s_q, \forall t \le l.$$

For *l* = 0, L^0 is empty and hence L_i^0 is empty for all *i*. Thus we have

$$\text{Bigsum}\left(0,\Delta,L_i^0\right) = 0,\ i = 1,2,\cdots,m.$$

This means that it is true for all *i*.

Now we will show the claim for (*i*, *l*) is true under the assumptions that the claim is true for (*i*–1, *l*) and (*i*, *l*–1). We prove it according to the following two cases:

Case 1. $c_i > 0$. In this case, any job *J* with release time *r* and size *p* satisfying $rs_i + p \le m_i = \Delta s_i$ in L_{i-1}^l cannot be passed over machine M_i to machine M_{i+1}. Hence we have

$$L_i^l = \left\{ j \mid r_j s_i + p_j > \Delta_l s_i, j \in L^l \right\}.$$

Thus we have

$$\text{Bigsum}\left(0,\Delta_l,L_i^l\right) = \text{Bigsum}\left(s_i,\Delta_l,L^l\right) \le \Delta_l\text{Bigsum}(s_i) = \Delta_l\sum_{q=i+1}^{m} s_q.$$

By the assumption on the claim for (*i*, *l*–1), we get

$$\sum_{t=0}^{l-1} \text{Bigsum}\left(0,\Delta_t,L_i^t\right) \le \left(\sum_{t=0}^{l-1}\Delta_t\right)\left(\sum_{q=i+1}^{n} s_q\right).$$

Adding up the above two inequality we get the claim for (i, l).

Case 2. $c_i \le 0$. In this case, we consider the sum of job sizes that assigned on machine M_i from phase 0 to phase l, which can be expressed as

$$\sum_{t=0}^{l}\left[\text{Bigsum}\left(0,\Delta_t,L_{i-1}^t\right)-\text{Bigsum}\left(0,\Delta_t,L_i^t\right)\right]$$

Because $c_i \le 0$, H_l, the height of machine M_i at the end of phase l, is at least

$$\begin{aligned}H_i &\ge \sum_{t=0}^{l} c_i^t = 2\Delta_0 s_i + \left(2-\frac{1}{r}\right)\Delta_1 s_i + \cdots + \left(2-\frac{1}{r}\right)\Delta_l s_i \\ &= 2\Delta_0 s_i + \left(2\Delta_1-\Delta_0\right)s_i + \cdots + \left(2\Delta_l - \Delta_{l-1}\right)s_i \\ &= \Delta_l s_i + s_i\sum_{t=0}^{l}\Delta_t\end{aligned}$$

By the rules of the algorithm, no job has release time greater than Δ_l. This means that there is no idle time in time interval $\left[\Delta_l,\Delta_l+\sum_{t=0}^{l}\Delta_t\right]$ on machine M_i. Hence the sum of the job sizes that assigned on machine M_i from phase 0 to phase l satisfies:

$$\sum_{t=0}^{l}\left[\text{Bigsum}\left(0,\Delta_t,L_{i-1}^t\right)-\text{Bigsum}\left(0,\Delta_t,L_i^t\right)\right] \ge s_i\sum_{t=0}^{l}\Delta_t .$$

By the inductive hypothesis for (i–1, l), we have

$$\sum_{t=0}^{l}\text{Bigsum}\left(0,\Delta_t\right) \le \left(\sum_{q=i}^{m} s_q\right)\left(\sum_{t=0}^{l}\Delta_t\right).$$

The above two inequalities include the truth of the claim for (i, l).

Lemma 2 show that, for every phase t, we have

$$\text{Bigsum}\left(0,\Delta_t,L_m^t\right)=0 .$$

This includes that L_m^t is empty for every phase t.

Theorem 3. The competitive ratio of algorithm R is not greater than 7.4641.

Proof: Suppose that the algorithm ended at phase k. Then the optimal value is at least $\Delta_{k-1} = r^{k-1}\Delta_0$ and the completion time of the algorithm is at most

$$\begin{aligned}\frac{\sum_{t=0}^{k} c_i^t}{s_i} &= 2\Delta_0 + \left(2-\frac{1}{r}\right)\Delta_1 + \cdots + \left(2-\frac{1}{r}\right)\Delta_k + \Delta_k \\ &= 2\Delta_0 + \left(2\Delta_1-\Delta_0\right) + \cdots + \left(2\Delta_k-\Delta_{k-1}\right) + \Delta_k \\ &= 2\Delta_k + \sum_{t=0}^{k}\Delta_t = \left(2r^k + \sum_{t=0}^{k} r^t\right)\Delta_0 \\ &= \sum_{t=0}^{k}\Delta_t = \left(2r^k + \frac{r^{k+1}-1}{r-1}\right)\Delta_0 .\end{aligned}$$

Hence the performance ratio is not greater than

$$\left(2r^k + \frac{r^{k+1}-1}{r-1}\right)\Bigg/ r^{k-1} < 2r + \frac{r^2}{r-1} = 3r + 1 + \frac{1}{r-1}.$$

It is easy to see that the best value of r is $1+\frac{\sqrt{3}}{3}$ and the performance ratio is $4+2\sqrt{3}\approx 7.4641$.

4. Conclusion

In this paper, on-line scheduling problem for jobs with arbitrary release times on uniform machines is considered. We developed an algorithm with the competitive ratio of 7.4641 which is better than existing result of 12. In order to improve the competitive ratio more detailed consideration should be taken in.

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 the Foundation Grant of Education Department of Hunan (No. 16A126).

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Cho, Y. and Sahni, S. (1980) Bounds for List Schedules on Uniform Processors. *SIAM Journal on Computing*, **9**, 91-103. https://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. https://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. https://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. https://doi.org/10.1145/258128.258201

[5] Berman, P., Charikar, M. and Karpinski, M. (2000) On-Line Load Balancing for Related Machines. *Journal of Algorithms*, **35**, 108-121. https://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. https://doi.org/10.1137/S0097539799527969

[7] Cheng, T.C.E., Ng, C. and Kotov, V. (2006) A New Algorithm for Online Uniform-Machine Scheduling to Minimize the Makespan. *Information Processing Letters*, **99**, 102-105.

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

[9] 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. https://doi.org/10.1016/j.ejor.2005.06.061

[10] Li, K., Zhang, H., Cheng, B. and Pardalos, P. (2018) Uniform Parallel Machine Scheduling Problem with Fixed Machine Cost. *Optimization Letter*, **12**, 73-86. https://doi.org/10.1007/s11590-016-1096-3

[11] Yin, Y., Chen, Y., Qin, K. and Wang, D. (2019) Two-Agent Scheduling on Unrelated Parallel Machines with Total Completion Time and Weighted Number of Tardy Jobs Criteria. *Journal of Scheduling*, **22**, 315-333. https://doi.org/10.1007/s10951-018-0583-z

[12] Cheng, X., Li, R. and Zhou, Y. (2016) On-Line Scheduling for Jobs with Arbitrary Release Times on Parallel Related Uniform Machines. *Intelligent Information Management*, **8**, 98-102. https://doi.org/10.4236/iim.2016.84008