Far East Journal of Applied Mathematics
Volume 74, Number 2, 2013, Pages 115-124

Published by Pushpa Publishing House, Allahabad, INDIA

LIST SCHEDULING FOR JOBS WITH ARBITRARY RELEASE TIMES AND UNIT-PROCESSING TIME

Jihuan Ding

College of Management Science
Qufu Normal University
P. R. China
e-mail: dingjihuan@hotmail.com

Abstract

We investigate the problem of on-line scheduling for jobs with arbitrary release times on m identical parallel machines. The goal is to minimize the makespan. For a special case that all the jobs have unit-processing time, we prove that algorithm LS has a tight bound of $3/2$ for general m machines.

1. Introduction

On-line scheduling has received great attention in decades. The most basic model is the classical on-line scheduling problem on m identical parallel machines which was proposed by Graham [3]. In the classical on-line scheduling problem, jobs arrive one by one (or over a job list), we do not know any information about the job list in advance, whenever a job arrives, it must be scheduled immediately on one of the machines without knowledge

---

2010 Mathematics Subject Classification: 90B35.

Keywords and phrases: on-line scheduling, makespan, competitive ratio, identical parallel machines.

Communicated by K. K. Azad

Received December 23, 2012

of any future jobs. Only after the current job is scheduled, the next job appears. The objective is to minimize the makespan. It is well known that the first on-line algorithm in scheduling theory, namely List Scheduling (LS) algorithm, for the problem was proposed by Graham [3].

Note that in the above model all jobs in the job list have release times zero. A more general version of the classical on-line scheduling problem on *m* identical parallel machines was given by Li and Huang [1]. In the general version, all jobs appear in form of orders at time zero. When a job appears, the scheduler is informed of the release time and processing time of the job. The problem can be formally defined as follows. A list of jobs is to be scheduled on *m* identical parallel machines, $M_1, M_2, ..., M_m$. We assume that all jobs appear on-line in a job list. Whenever a job J_j with release time r_j and processing time p_j appears, the scheduler has to assign a machine and a processing slot for J_j irrevocably without knowledge of any future jobs. The goal is to minimize the makespan. In this general on-line situation, the jobs' release times are assumed arbitrary, whereas in the existing literature the jobs' release times are normally non-decreasing. That is to say, although a job may appear first in the job sequence, but its release time may be greater than the release time of the job which appears later in the job sequence. It is obvious that the above model is just the classical on-line scheduling problem on *m* identical parallel machines if all the jobs have release time zero. The problem is called *on-line scheduling problem* for jobs with arbitrary release times [2].

The quality of an on-line algorithm A is usually measured by its competitive ratio

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

where $C_{\max}^A(L)$ and $C_{\max}^*(L)$ denote the makespans of the schedule produced by algorithm A and an optimal off-line algorithm, respectively.

Li and Huang considered the on-line scheduling problem for jobs with arbitrary release times first. For the general model of the problem where job processing times are arbitrary, Li and Huang [1] gave a lower bound of 2, and then showed an algorithm LS with tight bound of $3 - 1/m$. A modified algorithm MLS is also proposed which is better than LS for any $m \geq 2$, and the competitive ratio of MLS is bounded by 2.9392 for any $m \geq 2$. For the special model where the job length is in $[p, rp]\ (r \geq 1)$, the performance of algorithm LS is analyzed in [2]. They first gave an upper bound of

$$R(m, LS) = \begin{cases} 3 - \dfrac{1}{m} - \dfrac{1}{r}, & r \geq \dfrac{m}{m-1} \\ 1 + \dfrac{2r}{1+2r} + \dfrac{(m-1)r}{m(1+2r)}, & 1 \leq r < \dfrac{m}{m-1} \end{cases}$$

for general m and showed that the tight bound for $m = 1$ is $1 + \frac{r}{1+r}$. When $m = 2$, they presented a tight bound of the competitive ratio $1 + \frac{5r+4}{2(r+2)}$ for $r \geq 4$. For $r < 4$, they gave a lower bound and showed that 2 provides an upper bound for the competitive ratio.

Our results. In this paper, we consider the on-line scheduling problem for jobs with arbitrary release times. For a special model that all the jobs have unit-processing time, we prove that algorithm LS has a tight bound of $3/2$ for general m machines.

2. LS Algorithm for Jobs with Unit-processing Time

In this part, we give a tight bound of Algorithm LS for a special case that all the jobs have unit-processing time. In the following we will describe algorithm LS which is defined in Li and Huang [1]. Essentially, the algorithm assigns a job to be processed as early as possible when its order arrives.

Definition 2.1. Suppose that J_j is the current job with release time r_j and processing time 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 conditions:

1. Machine M_i is currently idle in interval $[T_1, T_2]$ and a job has been assigned on M_i to start processing at T_2.
2. $T_2 - \max\{T_1, r_j\} \geq 1$.

Algorithm LS

1. Assume that L_i is the scheduled completion time of machine $M_i (i = 1, 2, ..., m)$. Reorder machines so that $L_1 \leq L_2 \leq \cdots \leq L_m$ and let J_n be a new job given to the algorithm with release time r_n and running time $p_n = 1$. Let $t = \max\{r_n, L_1\}$.

2. If there exist some machines which have idle intervals for job J_n, select a machine M_i which has an idle interval $[T_1, T_2]$ for job J_n with minimal T_1. Then we start job J_n on machine M_i at time $\max\{T_1, r_n\}$ in the idle interval. Otherwise, we assign job J_n to machine M_1 to start the processing at time t.

In order to give the proof of the result, we first give some explanation of symbols. $[s]$: the integral part of s; (s): the fractional part of s; $\lceil s \rceil$: the smallest integer that is not smaller than s.

The following lemma gives an estimate of the number of jobs completed in time interval $[0, T]$ on one machine in schedule of algorithm LS on the assumption that there is no idle time interval with length equal to or greater than 1.

Lemma 2.1. *Let N denote the number of jobs completed on one machine in time interval* $[0, T]$. *If the length of all the idle intervals in schedule of algorithm LS is smaller than* 1, *then we have* (1) $N \geq \frac{T}{2}$ *if T is even*; (2) $N \geq \frac{T-1}{2}$ *if T is odd and there is one job with start time smaller than T* *which is completed after time T.* (3) $N \geq \frac{T+1}{2}$ *if T is odd and there is no job with start time smaller than T which is completed after time T.*

Proof. At most $N+1$ idle intervals are generated by the process of N jobs in $[0, T]$ on one machine. Then the total length of these idle intervals is smaller than $N+1$. So we have $(N+1)+N > T-1$. From the inequality we can get $N > \frac{T-2}{2}$. Therefore $N \geq \frac{T}{2}$ if T is even;

If T is odd and there is one job with start time smaller than T which is completed after time T. Then we also have $(N+1)+N > T-1$. So we can get $k \geq \frac{T-1}{2}$.

If T is odd and there is no job with start time smaller than T which is completed after time T. Then we have $(N+1)+N > T$, namely, $N \geq \frac{T+1}{2}$.

Theorem 2.1. *The competitive ratio of LS is* $R_{LS} = 3/2$.

Proof. We assume $L = \{J_1, J_2, ..., J_n\}$ is an arbitrary job list. Let $C_{\max}^{LS}(L)$ and $C_{\max}^{*}(L)$ denote the makespans of the schedule produced by algorithm LS and an optimal schedule, respectively. Without loss of generality, we assume J_n is the last job completed in schedule of algorithm LS and J_n is the only job with completion time $C_{\max}^{LS}(L)$. Let s_n be the start time of job J_n in schedule of algorithm LS. If $s_n = r_n$, then it is obvious that $R_{LS} \leq 3/2$. If $s_n > r_n$, then job J_n must be assigned on machine M_1 to start at time $s_n = L_1$ by the algorithm. In schedule of algorithm LS, let s be the least time point from which all the m machines are busy to L_1, and let k be the number of jobs (including job J_n) completed after time s on machine M_1. Then it is obvious that at least $(k-1)$ jobs are completed after time s on each of the other $m-1$ machines. So at least

$m(k-1)+1$ jobs in total are completed after time *s* in schedule of algorithm LS. We have $C_{\max}^{LS}(L) \le s+k$ and $C_{\max}^{*}(L) \ge s+1$, because at least one job has release time *s* by the algorithm.

We first consider the case that all the idle intervals in schedule of algorithm LS have length smaller than 1.

Case 1. [*s*] is even. By Lemma 2.1, it is easy to know the optimal schedule completes at most $m \cdot \frac{[s]}{2}$ more jobs before time [*s*] than the schedule of algorithm LS does.

Case 1.1. If $2k \le s+3$, then we have:

$$\frac{C_{\max}^{LS}(L)}{C_{\max}^{*}(L)} \le \frac{s+k}{s+1} \le \frac{2s+s+3}{2(s+1)} = \frac{3}{2}.$$

Case 1.2. If $2k \ge s+3 = [s]+2+1+(s)$, namely, $2k \ge s+4$. Then at least $m(k-1)+1-m\cdot\frac{[s]}{2}$ jobs are processed after time [*s*] in optimal schedule. So we have

$$C_{\max}^{*}(L) \ge [s] + \left\lceil \frac{1}{m}\left[m(k-1)+1-m\cdot\frac{[s]}{2} \right\rfloor \right\rceil$$

$$= \frac{1}{2}[s]+k.$$

Therefore:

$$\frac{C_{\max}^{LS}(L)}{C_{\max}^{*}(L)} \le \frac{s+k}{\frac{1}{2}[s]+k} \le \frac{2s+[s]+4}{2([s]+2)}$$

$$= \frac{3([s]+2)+2(s)-2}{2([s]+2)}$$

$$\le \frac{3}{2}.$$

Case 2. $[s]$ is odd.

Case 2.1. If $2k \leq s + 3$, then

$$\frac{C_{\max}^{LS}(L)}{C_{\max}^{*}(L)} \leq \frac{s + k}{s + 1} \leq \frac{2s + s + 3}{2(s + 1)} = \frac{3}{2}.$$

Case 2.2. If $2k > s + 3 = [s] + 2 + 1 + (s)$, namely, $2k \geq s + 5$. By Lemma 2.1, one machine in optimal schedule completes at most $\frac{[s] + 1}{2}$ more jobs before $[s]$ than it does in schedule of algorithm LS, and at least one machine in optimal schedule (the machine is idle immediately before s) completes at most $\frac{[s] - 1}{2}$ more jobs before $[s]$ than it does in schedule of algorithm LS. So at least $[m(k - 1) + 1] - \left[(m - 1)\frac{[s] + 1}{2} + \frac{[s] - 1}{2}\right]$ jobs must be processed after time $[s]$ in optimal schedule. Hence, we can derive

$$C_{\max}^{*}(L) \geq [s] + \left\lceil \frac{[m(k - 1) + 1] - \left[(m - 1)\frac{[s] + 1}{2} + \frac{[s] - 1}{2}\right]}{m} \right\rceil$$

$$= [s] + (k - 1) - \frac{[s] + 1}{2} + 1$$

$$= \frac{1}{2}[s] + k - \frac{1}{2}.$$

Case 2.2.1. If $(s) \leq \frac{1}{2}$, then

$$\frac{C_{\max}^{LS}(L)}{C_{\max}^{*}(L)} \leq \frac{s + k}{\frac{1}{2}[s] + k - \frac{1}{2}}$$

$$\leq \frac{2s + [s] + 5}{[s] + [s] + 5 - 1}$$

$$= \frac{3([s] + 2) + 2(s - [s]) - 1}{2([s] + 2)}$$

$$\leq \frac{3}{2}.$$

Case 2.2.2. $(s) > \frac{1}{2}$. In this case we need to estimate the number of jobs completed after time *s* in schedule of algorithm LS more carefully.

Case 2.2.2.a. The time point *s* appears exactly on machine M_1, namely, M_1 is idle immediately before time *s*, and *p* jobs with start time smaller than [*s*] are completed after time *s* in schedule of algorithm LS. For the *p* machines that process such *p* jobs, *k* jobs are completed after time *s* on each of them in schedule of algorithm LS. By Lemma 2.1, each of the *p* machines in optimal schedule completes at most $\frac{[s]+1}{2}$ more jobs before time [*s*] than it does in schedule of algorithm LS and each of the other $m - p$ machines in optimal schedule completes at most $\frac{[s]-1}{2}$ more jobs before time [*s*] than it does in schedule of algorithm LS. So we have

$$C_{\max}^{*}(L) \geq [s] + \left\lceil \frac{1}{m}\left\{[m(k-1) + p + 1] - \left[p \cdot \frac{[s]+1}{2} + (m-p)\frac{[s]-1}{2}\right]\right\}\right\rceil$$

$$= [s] + (k-1) - \frac{[s]-1}{2} + 1$$

$$= \frac{1}{2}[s] + k + \frac{1}{2}.$$

Therefore

$$\frac{C_{\max}^{LS}(L)}{C_{\max}^{*}(L)} \leq \frac{s+k}{\frac{1}{2}[s] + k + \frac{1}{2}} \leq \frac{2s + [s] + 5}{2([s]+3)}$$

$$= \frac{3([s]+3) + 2(s-[s]) - 4}{2([s]+3)}$$

$$\leq \frac{3}{2}.$$

Case 2.2.2.b. The time point *s* does not appear on machine M_1 and the start time of the first completed job after time *s* on machine M_1 is not greater than [*s*]. Then we have

$$C_{\max}^{LS}(L) \leq k + [s].$$

Hence

$$\frac{C_{\max}^{LS}(L)}{C_{\max}^{*}(L)} \le \frac{[s]+k}{\frac{1}{2}[s]+k-\frac{1}{2}}$$

$$\le \frac{2[s]+[s]+5}{[s]+[s]+5-1}$$

$$= \frac{3([s]+2)-1}{2([s]+2)}$$

$$\le \frac{3}{2}.$$

Case 2.2.2.c. The time point s does not appear on machine M_1, and p jobs with start time smaller than $[s]$ are completed after time s in schedule of algorithm LS, and the start time of the first completed job after time s on machine M_1 is greater than $[s]$. Similar to the analysis in Case 2.2.2.a, we have

$$C_{\max}^{*}(L) \ge [s] + \left\lceil \frac{1}{m}\left\{[m(k-1)+p+1] - \left[p\cdot\frac{[s]+1}{2} + (m-p)\frac{[s]-1}{2}\right]\right\}\right\rceil$$

$$= [s] + (k-1) - \frac{[s]-1}{2} + 1$$

$$= \frac{1}{2}[s] + k + \frac{1}{2}.$$

So

$$\frac{C_{\max}^{LS}(L)}{C_{\max}^{*}(L)} \le \frac{s+k}{\frac{1}{2}[s]+k+\frac{1}{2}} \le \frac{2s+[s]+5}{2([s]+3)}$$

$$= \frac{3([s]+3)+2(s-[s])-4}{2([s]+3)}$$

$$\le \frac{3}{2}.$$

Secondly, if there exist idle time intervals with length not smaller than 1 in schedule of algorithm LS, then we select such idle time interval with latest over time, say b, and choose a time point a such that $b - a = 1$. By the algorithm LS, all the jobs with start time greater than a must have release time greater than a also. So similar to the analysis in the first part of the proof, we can get $R_{LS} \leq \frac{3}{2}$.

Finally, the following instance shows that the bound of algorithm LS is tight. The instance consists $2m$ jobs in total, the first m jobs have release time $1 - \varepsilon$ and the last m jobs have release time zero. It is easy to know the makespan of algorithm LS for the instance is $3 - \varepsilon$, but the optimal makespan is 2. So $R_{LS} \geq \frac{3 - \varepsilon}{2}$, let ε tend to zero, we get the bound is tight.

3. Final Remarks

We consider the problem of scheduling for jobs with arbitrary release times on m identical parallel machines. For a special model where all the jobs have processing time 1, we derived that algorithm LS has a tight bound of $3/2$. Furthermore, some other special models of the problem are also worth to consider in future.

References

[1] R. Li and H. C. Huang, On-line scheduling for jobs with arbitrary release times, Computing 73 (2004), 79-97.

[2] R. Li and H. C. Huang, List scheduling for jobs with arbitrary release times and similar lengths, J. Sched. 10 (2007), 365-373.

[3] R. L. Graham, Bounds on multiprocessing timing anomalies, SIAM J. Appl. Math. 17 (1969), 416-429.