LS Algorithm for Semi-online Scheduling Jobs with Nondecreasing Release Times and Nondecreasing Processing Times

F. Tang
School. of Industry
Hunan Normal University
Changsha 410081, China

J. Nie
Dept. of Math
Hunan Normal University
Changsha 410081, China

***Abstract*—In this paper, semi-online scheduling jobs with non-decreasing release times and non-increasing processing times on m identical parallel machines is considered. The aim is to minimize the last completion time of all machines. It is proved that, for any m identical parallel machines, 3/2-1/2m is an upper bound of the worst case performance ratio of List Scheduling(LS) algorithm.**

Keywords-release time; processing time; semi-online; LS algorithm

I. INTRODUCTION

The problem of scheduling jobs on identical parallel machines is an important part of combinatorial optimization problem. It is defined as follows: Given a job set L = {J1,J2,...,Jn} of n jobs where job Ji has non-negative processing time pi, partition the job set into m subsets so as to minimize the maximum sum of processing times of the jobs in each subset.

A scheduling problem is called off line if we have complete information about the job data before constructing a schedule. In contrast, the scheduling problem is called online if the jobs appear one by one and it requires scheduling of the arriving job irrevocably on a machine without knowledge of the future jobs. The processing time of next job becomes available only after the current job is scheduled. The worst case performance ratio of an algorithm A is defined as:

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

where $C_{\max}^{A}(L)$ and $C_{\max}^{OPT}(L)$ are the maximum completion times of algorithm A and the optimal off-line algorithm respectively. Graham (1996) proposed the List Scheduling (LS) algorithm to minimize the maximum completion time for online scheduling of n jobs on m identical parallel machines. The LS algorithm always assign the current job to the machine that will complete it first.

In the Graham's classical on-line scheduling problem on m identical machines, all jobs are released at time zero one by one. Li and Huang(2004) generalized Graham's problem by assume that each job has a release time. A job Jj is informed of a 2-tuple (rj,pj), where rj and pj represent the release time and the processing time of the job Jj, respectively. This problem can be referred to as generalized on-line scheduling problem or on-line scheduling problem for jobs with arbitrary release times or orders on-line scheduling problem. Li & Huang showed that 3-1/m is the worst case performance ratio of the LS algorithm.

Liu et al. proposed semi-online schedule in 1996. Seiden et al. considered semi-online scheduling problem for jobs with release time zero and non-increasing processing times in 2000. They firstly showed that algorithm LS has worst case performance ratio 4/3-1/3m. When m=2, they proved that LS algorithm is the best. Later Cheng et al.(2012) proved that LS algorithm is also optimal for m=3 and they also proposed an algorithm with worst case performance 5/4 for m ≤ 4.

Li et al(2013) considered semi-online scheduling on m machines for jobs with non-decreasing release times and non-increasing processing times. They point out that it is easy to show that 3/2-1/2m is the upper bound of the worst case ratio of LS algorithm. But in fact the result is not so obvious. In this paper we give a new and full proof.

II. SYMBOLS

For job list $L=\{J_1,J_2,...,J_n\}$, we always use r_j and p_j to denote the release time and processing time of job J_j(j=1,2,…,n) and assume they satisfy the following inequalities:

$r_1 \le r_2 \le \ldots \le r_n$, $p_1 \ge p_2 \ge \ldots \ge p_n$.

Now we define some symbols as follows：

1. Idle interval (a,b): Let $(a_1,b_1),\ldots,(a_T,b_T)$ are all of the idle intervals of job list L in the LS schedule. (a,b) is one of the T idle intervals satisfying $b=\max_{1\le i\le p} b_i$;

2. U, U^*: $U = \sum_{i=1}^{p}(b_i - a_i)$; U^* denotes the total amount of idle time in an optimal schedule;

3. $J_{A(i,j)}$: denote the jth job assigned on machine M_i by algorithm A;

4. $s(A,J_{A(i,j)})$: denote the starting time of job $J_{A(i,j)}$ assigned by algorithm A;

5. l_i: the number of jobs assigned on machine M_i by algorithm LS;

6. $B_i(i=1,\ldots,m)$: $B_i=\{J_{LS(i,j)} | S(LS, J_{LS(i,j)})<b<S(LS, J_{LS(i,j)})+p_{LS(i,j)}\}$,i.e., the job $J_{LS(i,j)}$ in B_i is assigned to start before b and finish after b on machine M_i by the LS algorithm. It is obvious that $|B_i|\le 1$ holds;

7. Δ_i (i=1,…,m) and Δ: $\Delta=\max\{\Delta_1,\ldots,\Delta_m\}$, where if $B_i=\varnothing$, then $\Delta_i=0$ and if $B_i\neq\varnothing$ then $\Delta_i=\min\{S(LS, J_{LS(i,j)})+ p_{LS(i,j)}-b, S(LS, J_{LS(i,j)})- r_{LS(i,j)} \mid J_{LS(i,j)}\in B_i\}$

for (i=1,…,m).

8. Δ_{ij}^A: denote the amount of idle time between *(j-1)*th job and *j*th job assigned on machine M_i by algorithm A, i.e.

$$\Delta_{i1}^A = s\left(A, J_{A(i,1)}\right);$$

$$\Delta_{ij}^A = s\left(A, J_{A(i,j)}\right) - \left(s\left(A, J_{A(i,j-1)}\right) + p_{A(i,j-1)}\right). \qquad j \geq 2,\ i = 1,2,\cdots,m.$$

III. Main Conclusion and Its Proof

Theorem 1 For any job list $L=\{J_1,J_2,\ldots,J_n\}$, we have:

$$\frac{C_{\max}^{LS}(L)}{C_{\max}^{OPT}(L)} \leq \frac{3}{2} - \frac{1}{3m} \tag{1}$$

Proof: If (1) is not true, then there exist job list L(called counter example)satisfying

$$\frac{3}{2} - \frac{1}{3m} < \frac{C_{\max}^{LS}(L)}{C_{\max}^{OPT}(L)}.$$

We suppose that $L=\{J_1,J_2,\ldots,J_n\}$ is such a counter example that the number of jobs in L is least among all of the counter examples in the following discussion. We refer to a job list with the least number of jobs as minimal counter example. Let L_i denote the completion time of machine M_i just before J_n is assigned. Reorder the machines such that $L_1\leq L_2\leq\ldots\leq L_m$ holds. Then it is easy to see that $C_{\max}^{LS}(L) = L_1 + p_n$ holds by the minimality of job list L.

Case 1: There is no idle interval in the *LS* schedule.

In this case it is obvious that $\sum_{i=1}^{m} L_i + P_n = \sum_{i=1}^{n} p_i$ holds. Hence we have

$$\begin{aligned}
\frac{3}{2} - \frac{1}{2m} &< \frac{C_{\max}^{LS}(L)}{C_{\max}^{OPT}(L)} = \frac{m(L_1 + p_n)}{mC_{\max}^{OPT}(L)} \\
&\leq \frac{\sum_{i=1}^{m} L_i + mp_n}{mC_{\max}^{OPT}(L)} = \frac{\sum_{i=1}^{n} p_i + (m-1)p_n}{mC_{\max}^{OPT}(L)} \\
&\leq 1 + \frac{(m-1)p_n}{mC_{\max}^{OPT}(L)}.
\end{aligned}$$

Thus we get

$$2p_n > C_{\max}^{OPT}(L) \tag{2}$$

That means there is at most one job on each machine in any optimal schedule. Hence it is easy to get that the makespan of the *LS* algorithm is equal the makespan of any optimal algorithm. This is a contradiction.

Case 2: There is at least idle interval in the *LS* schedule.

By the definition of Δ_i, for any algorithm ρ, on each machine M_i the total amount of job processing time that is scheduled after time b_T in the LS schedule and can be moved forward before b_T to be processed by other algorithm is at most Δ_i.

Hence we have

$$U - \sum_{i=1}^{m} \Delta_i \leq U^*$$

Without loss of generality, we assume that machine M_r(1≤ r≤ m) is idle in the interval (a,b) in the LS schedule. Hence we get:

$$B_r = \Phi, \quad \Delta_r = 0, \quad \sum_{i=1}^{m} \Delta_i = \sum_{i=1, i\neq r}^{m} \Delta_i \leq (m-1)\Delta.$$

By the definition of (a,b), there exists job J_x($1\leq x\leq n$)satisfying $r_x=S(LS,J_x)=b$ and J_x is assigned on machine M_r by the LS algorithm. By $p_1\geq p_2\geq\ldots\geq p_n$ we have

$$C_{\max}^{OPT}(L) \geq r_X + p_X \geq b + p_n \tag{3}$$

By $\sum_{i=1}^{n} p_i + U^* \leq mC_{\max}^{OPT}(L)$, we get

$$\begin{aligned}
\frac{3}{2} - \frac{1}{2m} &< \frac{C_{\max}^{LS}(L)}{C_{\max}^{OPT}(L)} = \frac{m(L_1 + p_n)}{mC_{\max}^{OPT}(L)} \\
&\leq \frac{\sum_{i=1}^{m} L_i + mp_n}{mC_{\max}^{OPT}(L)} = \frac{\sum_{i=1}^{n} p_i + U + (m-1)p_n}{mC_{\max}^{OPT}(L)} \\
&= \frac{\sum_{i=1}^{n} p_i + U - \sum_{i=1}^{m-1} \Delta_i}{mC_{\max}^{OPT}(L)} + \frac{\sum_{i=1}^{m-1} \Delta_i + (m-1)p_n}{mC_{\max}^{OPT}(L)} \\
&\leq 1 + \frac{(m-1)(\Delta + p_n)}{mC_{\max}^{OPT}(L)}
\end{aligned}$$

Thus we get

$$2(\Delta + p_n) > C_{\max}^{OPT}(L) \tag{4}$$

If Δ=0, then we have eqn (2) holds. By the similar way used in **Case 1** we can conclude $C_{\max}^{OPT}(L) = C_{\max}^{LS}(L)$ holds. This is a contradiction. If Δ>0, we suppose that M_s satisfies:

$$B_S = \{J_{LS(S,K)}\}$$

$$\Delta = \Delta_s = \min\{S(LS, J_{LS(s,k)}) + p_{LS(s,k)} - b, S(LS, J_{LS(s,k)}) - r_{LS(s,k)}\} \tag{5}$$

By the rules of LS algorithm we have:

$$\{J_{LS(1,1)}, J_{LS(2,1)}, \cdots, J_{LS(m,1)}\} = \{J_1, J_2, \cdots, J_m\} \tag{6}$$

$$r_{LS(i,1)} = s(LS, J_{LS(i,1)})(i = 1,2,\cdots,m) \tag{7}$$

Lemma 1 If $\Delta \geq 0$ holds and Jt satisfies $S(LS, J_t) + p_t \leq b$ (i.e., job *Jt* is finished before *b*), then $P_{LS(S,k)} \leq p_t$ holds, where *s* and *k* are defined by eqn (5).

Proof: Suppose $P_{LS(S,k)} > p_t$ and *Jt* is assigned on *Mq*. By $P_{LS(S,k)} > p_t$ we can get $r_{LS(S,k)} \leq r_t$.

By the rules of LS algorithm, it is easy to see the following inequality holds: $S(LS, J_{LS(s,k)}) \leq S(LS, J_t)$,

$\because \Delta \leq S(LS, J_{LS(s,k)}) + p_{LS(s,k)} - b$, we have

$$\begin{aligned}
p_{LS(s,k)} &\geq b - S(LS, J_{LS(s,k)}) + \Delta \\
&\geq b - S(LS, J_t) + \Delta \\
&\geq p_t + \Delta \geq p_n + \Delta.
\end{aligned}$$

By $\Delta = \min\{S(LS, J_{LS(s,k)}) + p_{LS(s,k)} - b,$ $S(LS, J_{LS(s,k)}) - r_{LS(s,k)}\} > 0$, we have $r_{LS(s,k)} < S(LS, J_{LS(s,k)})$ That means $k \geq 2$, By our assumption, we have

$$\Delta + p_n \leq p_{LS(s,K)} \leq p_{LS(i,1)} (i = 1,2,\cdots,m)$$

In any optimal schedule, at least two job from $\{J_{LS(1,1)},\cdots,J_{LS(m,1)}, J_{LS(s,k)}\}$ are assigned on a machine together. Thus we get

$$2(\Delta + p_n) > C_{\max}^{OPT}(L) \geq 2p_{LS(s,k)} \geq 2(\Delta + p_n),$$

This is contradiction.

For any t, if $S(LS, J_t) + p_t \leq b$ holds, then by Lemma 1 and $\Delta \leq S(LS, J_{LS(s,k)}) + p_{LS(s,k)} - b < p_{LS(s,k)}$, we have,

$$p_t \geq \Delta \tag{8}$$

Corollary 1.1 If $\Delta > 0$, then there are at most two jobs which are finished before b on any machine.

Corollary 1.2 If there are two machines M_i, M_t $(1 \leq i,t \leq m)$ satisfying

$$S(LS, J_{LS(t,2)} + p_{LS(t,2)} \leq r(LS, J_{LS(i,1)}) + p_{LS(i,1)},$$

then M_i can not finish any job other than $J_{LS(i,1)}$ before b in the LS schedule.

Lemma 2 In the *LS* schedule, reorder $M_1, M_2,\ldots,M_m$ such that the following inequalities hold:

$r_{LS(1,1)}+p_{LS(1,1)} \leq r_{LS(2,1)}+p_{LS(2,1)} \leq \ldots \leq r_{LS(m,1)} + p_{LS(m,1)}$.

I. If $\{J_{LS(1,1)},\ldots,J_{LS(m,1)},J_{LS(1,2)},\ldots,J_{LS(m,2)}\}=\{J_1,\ldots,J_{2m}\}$, then we have $\sum_{i=1}^{m}(\Delta_{i1} + \Delta_{i2}) \leq U^*$

II. If $\{J_{LS(1,1)},\ldots,J_{LS(m,1)},J_{LS(1,2)},\ldots,J_{LS(m,2)}\} \neq \{J_1,\ldots,J_{2m}\}$ then there exists $w(1<w\leq m)$ satisfying $\sum_{i=1}^{m}\Delta_{i1} + \sum_{i=1}^{\omega-1}\Delta_{i2} \leq U^*$.

Proof: Please refer to the full paper. □

Now we will finish the proof of Case 2 of Theorem 1 according to the following two subcases:

Case2.1.

$$\{J_{LS(1,1)},\cdots,J_{LS(m,1)}; J_{LS(1,2)},\cdots,J_{LS(m,2)}\} = \{J_1, J_2,\cdots,J_{2m}\}.$$

In this case let

$$\Delta_i^* = \sum_{i=3}^{l_i}\Delta_{ij}^{LS}, \quad i = 1,2,\cdots,m, \tag{9}$$

where l_i denotes the number of jobs assigned on machine M_i in the LS scedule. By **Lemma 2** we have $\sum_{i=1}^{m}(\Delta_{i1}^{LS} + \Delta_{i2}^{LS}) \leq U^*$. By **Corollary 1.1** we get

$$\Delta_i^* = \Delta_{i3}^{LS}, i = 1,2,\cdots m \tag{10}$$

$$U = \sum_{i=1}^{m}(\Delta_{i1}^{LS} + \Delta_{i2}^{LS} + \Delta_{i3}^{LS}) = \sum_{i=1}^{m}(\Delta_{i1}^{LS} + \Delta_{i2}^{LS}) + \sum_{i=1}^{m}\Delta_{i3}^{LS} \tag{11}$$

Let s and k satisfy eqn (5). By the proof of Lemma 1 we know $k \geq 2$. By the definition of Δ we have

$$S(LS, J_{LS(s,k)}) - r_{LS(s,k)} > 0; \qquad S(LS, J_{LS(s,k)}) + p_{LS(s,k)} - b > 0.$$

By the definition of Δ_{ij}^A, $S(LS, J_{LS(s,k)}) = S(LS, J_{LS(s,k-1)}) + p_{LS(s,k-1)}$ is obvious. Thus we get

$$\Delta_{sk}^{LS} = 0. \tag{12}$$

If machine M_s just finishes job $J_{LS(s,1)}$ before b, then k=2.

$$\because S(LS, J_{LS(s,k)}) + p_{LS(s,k)} > b \quad \therefore \Delta_{st}^{LS} = 0 (t \geq 2) \quad \therefore \Delta_s^* = \Delta_{s3}^{LS} = 0$$

If machine M_s has finished two jobs $J_{LS(s,1)}$ and $J_{LS(s,2)}$ before b, then k=3. By eqn (10) and eqn (12) we have $\Delta_s^* = \Delta_{s3}^{LS} = 0$.

Thus $\sum_{i=1}^{m}\Delta_i^* = \sum_{i=1,i\neq s}^{m}\Delta_i^* = \sum_{i=1,i\neq s}^{m}\Delta_{i3}^{LS}$. By (11) we get

$$U = \sum_{i=1}^{m}(\Delta_{i1}^{LS} + \Delta_{i2}^{LS} + \Delta_{i3}^{LS}) = \sum_{i=1}^{m}(\Delta_{i1}^{LS} + \Delta_{i2}^{LS}) + \sum_{i=1,i\neq s}^{m-1}\Delta_i^*.$$

Let $\Delta^* = \max\{\Delta_1^*, \Delta_2^*, \cdots, \Delta_m^*\}$, then

$$\frac{3}{2} - \frac{1}{2m} < \frac{C_{\max}^{LS}(L)}{C_{\max}^{OPT}(L)} \leq \frac{\sum_{i=1}^{n} p_i + U}{mC_{\max}^{OPT}(L)} + \frac{(m-1)p_n}{mC_{\max}^{OPT}(L)}$$
$$= \frac{\sum_{i=1}^{n} p_i + \sum_{i=1}^{m}(\Delta_{i1}^{LS} + \Delta_{i2}^{LS})}{mC_{\max}^{OPT}(L)} + \frac{\sum_{i=1,i\neq s}^{m-1}\Delta_i^* + (m-1)p_n}{mC_{\max}^{OPT}(L)}$$
$$\leq 1 + \frac{(m-1)(\Delta^* + p_n)}{mC_{\max}^{OPT}(L)}$$

Therefore $2(\Delta^* + p_n) > C_{\max}^{OPT}(L)$. By eqn (4) we can get

$$2\min\{\Delta, \Delta^*\} + 2p_n > C_{\max}^{OPT}(L) \tag{13}$$

(1) If $\Delta^*=0$, then by the proof of Case 1 we have $C_{\max}^{OPT}(L) = C_{\max}^{LS}(L)$. This is a contradiction.

(2) $\Delta^*>0$, suppose machine $M_u(u\in\{1,2,\ldots, m.\})$. satisfies:

$$\Delta^* = \Delta_u^* = \Delta_{u3}^{LS} \tag{14}$$

M_u has finished two jobs $J_{LS(u,1)}$, $J_{LS(u,2)}$ before b. By eqn (8), $p_{LS(u,1)} \geq p_{LS(u,2)} > \Delta$ holds. Thus

$$C_{\max}^{OPT}(L) \geq b + p_n \geq p_{LS(u,1)} + p_{LS(u,2)} + \Delta^* + p_n$$
$$> \Delta + \Delta^* + p_n + p_n \geq 2\min\{\Delta, \Delta^*\} + 2p_n.$$

This contradicts to eqn (13).

Case 2.2.

$$\{J_{LS(1,1)},\cdots,J_{LS(m,1)}; J_{LS(1,2)},\cdots,J_{LS(m,2)}\} \neq \{J_1, J_2,\cdots,J_{2m}\}$$

In this case we firstly reorder machines $M_1,M_2,\ldots,M_m$ such that

$r_{LS(1,1)}+p_{LS(1,1)} \leq \ldots \leq r_{LS(m,1)}+p_{LS(m,1)}$. Then we can prove this case similarly as Case 2.1.

ACKNOWLEDGMENT

This work is supported by the Chinese National Natural Science Foundation Grant(No.11471110), Project supported by the Science and Technology Department of Hunan Province(2012GK3122)，and the Scientific Research Fund of Hunan Provincial Education Department(12C0198).

REFERENCES

[1] Cheng T.C.E., Kellerer H., Kotov V., Algorithms better than LPT for semi-online scheduling with decreasing processing times[J]. Operations Research Letters, Vol.40: pp349-352. 2012,

[2] Graham RL, Bounds on multiprocessing timing anomalies. SIAM J Appl. Math,Vol.17: pp416-429, 1969.

[3] Li R, Huang HC, On-line scheduling for jobs with arbitrary release times. Computing ,Vol.73: pp79-97, 2004.

[4] Li, R., Yang,L., He,X., Chen,Q., Cheng,X., Semi-online scheduling for jobs with release times, Journal of Combinatorial Optimization,Vol.26(3), pp448-464, 2013.

[5] Liu W. P., Sidney J. B., Vliet A., Ordinal algorithm for parallel machine scheduling. Operations Research Letters. Vol.18, pp223-232, 1996.

[6] Seiden S., Sgall J., Woeginger G., Semi-online scheduling with decreasing job sizes[J]. Operations Research Letters, Vol.27: pp215-222, 2000.