ISSN 1009 – 3095 *Journal of Zhejiang University SCIENCE* V.3, No.1, P.60 – 64, Jan. – Mar., 2002
http://www.periodicals.com.cn; http://www.zju.edu.cn/depart/zdxb

http://lib.zju.edu.cn/eindex.htm; http://www.zjupress.com; jzu_s@mail.hz.zj.cn

Algorithms for semi on-line multiprocessor scheduling problems*

HE Yong(何　勇)[1], YANG Qi-fan(杨启帆)[1], TAN Zhi-yi(谈之奕)[2], YAO En-yu(姚恩瑜)[1]

([1] *Department of Mathematics, Zhejiang University, Hangzhou 310027, China*)

([2] *Department of System Science & Engeering, Zhejiang University, Hangzhou 310027, China*)

Received Dec.18, 2000; revision accepted Mar.18, 2001

Abstract: In the classical multiprocessor scheduling problems, it is assumed that the problems are considered in off-line or on-line environment. But in practice, problems are often not really off-line or on-line but somehow in between. This means that, with respect to the on-line problem, some further information about the tasks is available, which allows the improvement of the performance of the best possible algorithms. Problems of this class are called semi on-line ones. The authors studied two semi on-line multiprocessor scheduling problems, in which, the total processing time of all tasks is known in advance, or all processing times lie in a given interval. They proposed approximation algorithms for minimizing the makespan and analyzed their performance guarantee. The algorithms improve the known results for 3 or more processor cases in the literature.

Key words: analysis of algorithm, on-line scheduling, worst-case ratio
Document code: A　　**CLC number:** TP393

INTRODUCTION

In the parallel identical machine scheduling problem, we are given a set $\mathscr{J} = \{p_1, p_2, \cdots, p_n\}$ of independent jobs, each with a positive processing time, that must be scheduled on m parallel and identical machines $M = \{M_1, M_2, \cdots, M_m\}$. We identify the jobs with their processing times. The jobs and machines are available at time zero, and no preemptionis allowed. The objective is to minimize the maximum machine completion time (make-span) $C_{\max}$. This NP-complete problem usually denoted by $P \| C_{\max}$, is a classical and basic problem in scheduling theory, and has many applications in practice (Graham, 1966). A scheduling problem is called on-line if it requires scheduling jobs irrevocably on the machines as soon as they are given, without any knowledge about jobs that follow later on. If we have full information on the job data before constructing a schedule, this problem is called off-line. In practice, problems are often not really on-line or off-line but somehow in between. This means that, some partial information about the jobs is available and we cannot rearrange any job which has been assigned to machines. Such a case is defined as semi on-line problem. Algorithms for (semi) on-line problem are called (semi) on-line algorithms. Different partially available information gives rise to different semi on-line problem.

In a worst-case analysis, the performance of an on-line algorithm is measured through the worst-case ratio with respect to the optimal solution of the off-line problem. For a set $\mathscr{J}$ of jobs and an approximation algorithm A, let w_A denote the makespan produced by the algorithm A and let w^* denote the optimal makespan. Then the worst-case ratio of the algorithm A is defined as

$$R_A = \sup_{\mathscr{J}}\left\{\frac{w_A}{w^*}\right\}.$$

The simplest algorithm for the on-line parallel machine scheduling problem is the List Scheduling (LS) algorithm introduced by Graham (1966). This algorithm always assigns the current job to the machine with minimum workload on it at the moment. Applying LS to $P \| C_{\max}$, Graham showed $R_{\mathrm{LS}} = 2\text{-}1/m$. Faigle et

* Project supported by the National Natural Science Foundation of China (Nos.19701028 and 19971078) and National 973 Research Project of China.

al. (1989) observed that it is the best possible on-line algorithm for 2 and 3 machines. It means that there is no deterministic on-line algorithm for $P \parallel C_{\max}$ with a worst-case ratio better than 3/2 and 5/3 for $m = 2$, 3, respectively. For $m \geqslant 4$, several algorithms were proposed which had slightly better worst-case ratio than LS in references (Albers, 1997; Bartal, 1992; Galambos and Woeginger, 1993; Karger et al., 1996). He and Zhang (1999) and He (2000) showed that LS is still the best possible if the processing times are tightly-grouped for any $m \geqslant 2$.

Recently, many authors showed that several different additional partially available bits of information admit the possibility of constructing a schedule with a smaller worst-case ratio than that of LS. For example, jobs are known to arrive in non-increasing order of their processing times but the processing time of each job is unknown beforeit is assigned (Liu et al., 1996), the total processing time of all jobs is known in advance (Kellerer et al., 1997), a buffer of length k is available to maintain k jobs when scheduling (Kellerer et al., 1997), the processing time of the largest job is known as a priori (He and Zhang, 1999), etc. It has been shown that, for each semi on-line problem above, the best possible algorithm has a worst-case ratio of 4/3 when $m = 2$. So LS is not the best possible semi on-line algorithm in these cases.

Although there are many studies on semi on-line secheduling, most of them are focused on the $m = 2$ machine case, the $m \geqslant 3$ cases are relatively unstudied. In this paper, we consider these cases for the following two semi on-line modelling: the total processing time of all jobs is known in advance, and all processing times lie in a given interval. We will present their respective algorithms and show their worst-case ratios.

In the following sections of the paper, $l(M_i)$ denotes the total processing time of the jobs assigned to M_i after assigning all jobs by an approximation algorithm A, hence $w_A = \max_{i=1,\cdots,m}\{l(M_i)\}$.

THE TOTAL SUM OF PROCESSING TIMES IS KNOWN

In this section, we consider a semi on-line problem of $P_m \parallel C_{\max}$, where we assume that the jobs arrive one by one and we have no more knowledge about the jobs except that the total sum of all processing times is known in advance. We devise an algorithm with a worst-case ratio of 3/2 for $m = 3$ machine case and then generalize it to $m > 3$ machine cases. Without loss of generalization, we suppose the total sum of all processing time is mC. In the remainder of the paper, define machine load to be the total processing time of the jobs already assigned to that machine during the algorithm procedure, call a machine a new machine if it did not process any job before that time, and say that a machine is closed (or close the machine) if it does not process any more job after that time.

The following algorithms H_1 and H_2 were devised to solve $m = 3$ machine case and the general $m > 3$ machine case respectively.

Algorithm H_1

Always assigns the incoming jobs to M_1 until there is a job p such that the new workload of M_1 is greater than $3C/2$ as long as it is assigned to M_1. Then assign p to machine M_2 and assign all remaining jobs to M_3.

Algorithm H_2

1. If the incoming job has a processing time of at least C, then assign it to a new machine, and close this machine.

2. If the incoming job has a processing time of less than C, then assign it to a machine with maximum current load as long as the new machine load is smaller than or equal to $(2m-3)C/(m-1)$. If the new machine load is greater than $(2m-3)C/(m-1)$ go to 3.

3. Assign the incoming job to a new machine. If the total processing time of all remaing jobs is no greater than $(2m-3)C/(m-1)$, assign them to the machine with the minimum current load, otherwise, go to 1.

Theorem 1. Algorithm H_1 has a tight worst-case ratio of 3/2.

Proof: It is clear that $w^* \geqslant C$, $w^* \geqslant \max_{i=1,\cdots,n}\{p_i\}$. From the rule of the algorithm H_1, we know that $l(M_1) \leqslant 3C/2 \leqslant 3w^*/2$. Since H_1 forces M_2 to process only one job, hence $l(M_1) \leqslant \max_{i=1,\cdots,n}\{p_i\} \leqslant w^*$. By the definition of p in the algorithm H_1, we know that $l(M_1) + l(M_2) \geqslant 3C/2$, hence $l(M_3) \leqslant$

$3C - (l(M_1) + l(M_2)) \leqslant 3C/2 \leqslant 3w^*/2$. Therefore we have $w_{H_1}/w^* \leqslant 3/2$. The instance $\{p_1 = 1/2, p_2 = 1, p_3 = 1/2, p_4 = 1\}$ can show that the ratio is tight. Thus we complete the proof.

As we know that for the $m = 3$ case, LS algorithm is the best possible on-line algorithm with a worst-case ratio 5/3, hence we deduce that H_1 is better than LS. It verifies that some partial knowledge about the jobs (i.e., the total sum) is of help for constructing an on-line algorithm with small worst-case ratio.

Theorem 2. Algorithm H_2 has a worst-case ratio no greater than $2 - 1/(m-1)$.

Proof: It is obvious that $w^* \geqslant C$, $w^* \geqslant \max_{i=1,\cdots,n}\{p_i\}$. If there are at least $m-1$ jobs with processing times no less than C, then H_2 assigns these jobs to different machines. Hence the total processing time of the remaining unprocessed jobs is no greater than C. It is clear that H_2 yields an optimal schedule and we have proven our case. If there are $m-2$ jobs with processing times no less than C, without loss of generalization, we suppose that every machine of $\{M_m, M_{m-1}, \cdots, M_3\}$ is assigned to process such a job by H_2. Since these machines do not process any other job, their machine loads are no greater than w^*. Suppose $p_t < C$ is the first job such that the machine load of M_1 is greater than $(2m-3)C/(m-1)$, then the algorithm assigns p_t to M_2. Furthermore, the total processing time of all remaining jobs is no greater than $mC - (m-2)C - (2m-3)C/(m-1) = C/(m-1)$. Since if all the remaining jobs are assigned to M_2, the new load of M_2 is less than $p_t + C/(m-1) \leqslant C + C/(m-1) \leqslant (2m-3)C/(m-1)$, Hence we know that $l(M_1) \leqslant (2m-3)C/(m-1) \leqslant (2m-3)w^*/(m-1)$, $l(M_2) \leqslant (2m-3)C/(m-1) \leqslant (2m-3)w^*/(m-1)$, which is the desired ratio.

In general, suppose there are i, $0 \leqslant i \leqslant m-3$ jobs with processing times no less than C. Without loss of generalization, we assume that every machine of $\{M_m, M_{m-1}, \cdots, M_{m-i+1}\}$ is assigned to process such a job by H_2. Since these machines do not process any other job, their machine loads are no greater than w^*. Suppose $p_t < C$ is the first job such that the machine loads of $M_1, \cdots, M_{m-i-2}$ are greater than $(2m-3)C/(m-1)$, then the algorithm assigns p_t to M_{m-i-1}. Because $p_t < C$, we know that the machine loads of $M_1, \cdots, M_{m-i-2}$ are at least $(2m-3)C/(m-1) - C = (m-2)C/(m-1)$ just before assigning p_t. Furthermore, the total processing time of all remaining jobs is no greater than

$$mC - iC - (m-i-3)\frac{(m-2)C}{m-1} - \frac{(2m-3)C}{m-1}$$
$$= \frac{(2m-i-3)C}{m-1} \leqslant \frac{(2m-3)C}{m-1}.$$

By the rule of H_2, all the remaining jobs after p_t are assigned to M_{m-i}, we know that $l(M_{m-i}) \leqslant (2m-3)C/(m-1)$. Since no new job is assigned to $M_1, \cdots, M_{m-i-1}$ after p_t, we have $l(M_j) \leqslant (2m-3)C/(m-1)$, $j = 1, 2, \cdots, m-i-1$. Hence we can conclude that after processing all jobs, every machine has a load no greater than $(2m-3)w^*/(m-1)$ and the proof is completed.

Since we know that LS algorithm has a worst-case ratio of $2 - 1/m$, we conclude that H_2 is still better than LS for any $m > 3$ by utilizing the knowledge of the total sum of all processing times. Note that Chen et al. (1995) showed that any online algorithm for solving $P_4 \parallel C_{\max}$ has a worst-case ratio of 1.73101, we know that H_2 is even better than the best possible on-line algorithm.

PROCESSING TIMES ARE IN A GIVEN INTERVAL

This section assumes that the jobs arrive one by one and the processing times of all jobs are in the interval $[p, rp]$, where $p > 0$, $r \geqslant 1$. We consider the semi on-line problem $P_m \parallel C_{\max}$. Since He and Zhang (1999) showed that LS is the best possible semi on-line algorithm for $P_2 \parallel C_{\max}$, and He (2000) also showed that if the processing times of all jobs are well tightly-grouped, LS is still the best, so we focus on the worst-case ratio of LS for the $m = 3$ machine case. The following Theorem 3 is cited from (He, 2000).

Theorem 3. Applying *LS* to the problem $P_m \parallel C_{\max}$, if all jobs have their processing times within the interval $[p, rp]$, where $p > 0$, $1 \leqslant r \leqslant m/(m-1)$, then the tight worst case ratio of *LS* is $1 + (m-1)(r-1)/m$, and *LS* is the best possible semi on-line algorithm in this interval.

The following result is the direct conclusion of Theorem 3.

Corollary 4. For $m = 3$, *LS* has a tight worst case ratio of $(2r+1)/3$ if $1 \leqslant r \leqslant 3/2$.

In the following, we normalize the processing times of all jobs such that $p = 1$.

Lemma 5. If the number of jobs is no greater than 6 and $r \leqslant 2$, then we have

$$\frac{w_{LS}}{w^*} \leqslant 1 + \frac{r-1}{2}. \tag{1}$$

Proof: For $n = 1, 2, 3$, it is clear that for such a number of jobs, *LS* yields an optimal solution. For $n = 4, 5, 6$, we consider them case by case. Without loss of generalization, we assume that p_n determines the *LS* makespan w_{LS}.

Case 1 $n = 4$. *LS* assigns the first 3 jobs to different machines. Without loss of generalization, we assume that $p_1 \leqslant p_2 \leqslant p_3$. Then we have

$$\frac{w_{LS}}{w^*} = \frac{p_1 + p_4}{p_1 + \min\{p_2, p_3, p_4\}} = 1 + \frac{p_4 - \min\{p_2, p_3, p_4\}}{p_1 + \min\{p_2, p_3, p_4\}} \leqslant 1 + \frac{r-1}{2}. \tag{2}$$

Case 2 $n = 5$. *LS* still assigns the first 3 jobs to different machines and we can assume that $p_1 \leqslant p_2 \leqslant p_3$. Suppose p_i is assigned to M_i, $i = 1, 2, 3$ by *LS*. Then *LS* assigns p_4 to M_1 by rule. From $r \leqslant 2$, we have that $p_1 + p_4 \geqslant p_2$. Hence p_5 must be assigned to M_2 by *LS* and $w_{LS} = p_2 + p_5$.

Now we consider the optimal schedule. It is clear that there is no machine M_j which processes at least 4 jobs. Since otherwise $w^* \geqslant 4 \geqslant 2r \geqslant w_{LS}$. If there is one machine M_j which processes exactly 3 jobs, then M_j must process one job in $\{p_2, p_3, p_5\}$. Hence $w^* \geqslant \min\{p_2, p_3, p_5\} + 2 = \min\{p_2, p_5\} + 2$. Combining it with $\max\{p_2, p_5\} \leqslant r \leqslant 2$, we know that $w_{LS} = p_2 + p_5 = \min\{p_2, p_5\} + \max\{p_2, p_5\} \leqslant \min\{p_2, p_5\} + 2 \leqslant w^*$. Hence we deduce that in the optimal schedule, there are two machines, each of which processes two jobs, and the third machine processes one job. Suppose machine M_j processes two jobs and one of the jobs is p_2, p_3 or p_5, then by combining $w^* \geqslant 2$, $w^* \geqslant \min\{p_2, p_3, p_5\} + 1 = \min\{p_2, p_5\} + 1$, and every job satisfies $1 \leqslant p_i \leqslant r$, we have

$$\frac{w_{LS}}{w^*} = 1 + \frac{w_{LS} - w^*}{w^*} \leqslant 1 + \frac{p_2 + p_5 - \min\{p_2, p_5\} - 1}{w^*} \leqslant 1 + \frac{r-1}{2}. \tag{3}$$

Case 3 $n = 6$. By the same analysis as Case 1-2, we conclude that $p_1 \leqslant p_2 \leqslant p_3$ and p_i is assigned to M_i, $i = 1, 2, 3$ by *LS*. Since p_6 determines w_{LS}, we have $w_{LS} = p_3 + p_6$. By similar arguments as in Case 2, we deduce that in the optimum, every machine processes exactly two jobs. Hence $w^* \geqslant p_3 + 1$, and

$$\frac{w_{LS}}{w^*} = 1 + \frac{w_{LS} - w^*}{w^*} \leqslant 1 + \frac{p_3 + p_6 - (p_3 + 1)}{w^*} \leqslant 1 + \frac{r-1}{2}. \tag{4}$$

This completes the proof.

Theorem 6. For $m = 3$, *LS* has a worst case ratio of at most $1 + 2r/9$ if $3/2 < r < 9/5$.

Proof: By an averange argument, we know that $w^* \geqslant \sum_{j=1}^{n} p_j/3$. Without loss of generalization, we assume that p_n determines the *LS* makespan w_{LS}. Denote s as the start time of p_n, then we have $s \leqslant \sum_{j=1}^{n-1} p_j/3$ by the *LS* rule. Therefore

$$\frac{w_{LS}}{w^*} = \frac{s + p_n}{w^*} \leqslant \frac{\sum_{j=1}^{n-1} p_j/3 + p_n}{w^*} = \frac{\sum_{j=1}^{n} p_j/3 + 2p_n/3}{w^*} \leqslant 1 + \frac{2p_n}{3w^*}. \tag{5}$$

If $p_n \leqslant rw^*/3$, then we have $w_{LS}/w^* \leqslant 1 + 2r/9$, which is the desired ratio. Hence we assume $p_n > rw^*/3$. Let k denote the largest inte-

ger no bigger than $n/3$, then there is at least one machine which processes at least k jobs in an optimal schedule. Since every job has a processing time at least 1, we have $w^* \geqslant k$. We thus have

$$r \geqslant p_n > \frac{rw^*}{3} \geqslant \frac{rk}{3}, \qquad (6)$$

that is to say $k < 3$. Hence $n \leqslant 6$. Combining Lemma 5 and $(r-1)/2 \leqslant 2r/9$, we have

$$\frac{w_{LS}}{w^*} \leqslant 1 + \frac{r-1}{2} \leqslant 1 + \frac{2r}{9}. \qquad (7)$$

This completes the proof.

Theorem 7. For $m = 3$, LS has a tight worst case ratio of $(1+r)/2$ if $9/5 \leqslant r \leqslant 2$.

Proof: By the same arguments as in Theorem 6, we know that inequality (5) is still valid. Hence if $p_n \leqslant 3(r-1)w^*/4$, we get $w_{LS}/w^* \leqslant (r+1)/2$, which is the desired ratio. Hence we assume $p_n > 3(r-1)w^*/4$. Let k denote the largest integer no bigger than $n/3$, then there is at least one machine which processes at least k jobs in an optimal schedule. Since every job has a processing time at least 1, we have $w^* \geqslant k$. We thus have

$$r \geqslant p_n > \frac{3(r-1)w^*}{4} \geqslant \frac{3(r-1)k}{4}, \qquad (8)$$

that is to say

$$k < \frac{4r}{3(r-1)} = \frac{4}{3}\left(1 + \frac{1}{r-1}\right) \leqslant \frac{4}{3}\left(1 + \frac{1}{9/5-1}\right) = 3. \qquad (9)$$

Applying Lemma 5, we have $w_{LS}/w^* \leqslant (r+1)/2$. The following instance can show the ratio is tight: $\{p_1 = p_2 = p_3 = 1,\ p_4 = r\}$.

Theorem 8. For $m = 3$, LS has a tight worst-case ratio of 5/3 if $r \geqslant 3$ and is the best possible semi on-line algorithm for $r \geqslant 6$.

Proof: For the upper bound, it is the direct conclusion of Graham (1966), the instance $\{p_1 = p_2 = \cdots = p_6 = 1, p_7 = 3\}$ can show its tightness. For $r \geqslant 6$, The instance sequences with jobs in $\{p_1 = p_2 = p_3 = 1,\ p_4 = p_5 = p_6 = 3,\ p_7 = 6\}$ (Faigle, 1989) can show that LS is the best possible semi on-line algorithm.

Unfortunately, we do not get better estimation of the worst-case ratio of LS for $2 \leqslant r \leqslant 3$ and do not know which algorithm is the best possible for $3/2 < r < 9/5$ and $2 < r < 6$. We conjecture that LS is still the best possible for these values of r. To reach the goal, new technique should be introduced.

References

Albers, S., 1997. Better bounds for on-line scheduling. Proc. 29th Annual ACM Symp. on Theory of Computing, p.130 – 139.

Bartal, Y., Fiat, A., Karloff, A., et al., 1992. New algorithm for an ancient scheduling problem. Proc. 24th Annual ACM Symp. on Theory of Computing, p.51 – 58.

Chen, B., Vliet, A. van, Woeginger, G., 1995. New lower and upper bounds for on-line scheduling. *Oper. Res. Letters*, **18**: 127 – 131.

Faigle, U., Kern, W., Turán, G., 1989. On the performance of on-line algorithm for particular problems, *Acta Cybernetica*, **9**: 107 – 119.

Galambos, G., Woeginger, G., 1993. An on-line scheduling heuristic with better worst-case ratio than Graham's list scheduling. *SIAM J. on Computing*, **22**: 349 – 355.

Graham, R. L., 1966. Bounds for certain multiprocessing anomalies. *Bell System Tech.*, **45**: 1563 – 1581.

He, Y., 2000. The optimal online parallel machine scheduling. *Computers & Mathematics with Applications*, **39**: 117 – 121.

He, Y., Zhang, G., 1999. Semi on-line scheduling on two identical machines. *Computing*, **62**: 179 – 187.

Karger, D. R., Phillips, S. J., Torng, E., 1996. A better algorithm for an ancient scheduling algorithm. *J. of Algorithms*, **20**: 400 – 430.

Kellerer, H., Kotov, V., Speranza, M., et al., 1997. Semi on-line algorithms for the partition problem. *Oper. Res. Letters*, **21**: 235 – 242.

Liu, W. P., Sidney, J. B., Vliet, A. van, 1996. Ordinal algorithms for parallel machine scheduling. *Oper. Res. Letters*, **18**: 223 – 232.