Journal of Zhejiang University SCIENCE
ISSN 1009-3095
http://www.zju.edu.cn/jzus
E-mail: jzus@zju.edu.cn

Semi on-line scheduling for maximizing the minimum machine completion time on three uniform machines

LUO Run-zi (罗润梓), SUN Shi-jie (孙世杰)

(*Department of Mathematics, Shanghai University, Shanghai 200444, China*)

E-mail: luo_rz@eyou.com; sun_sj@eyou.com

Received Feb. 30, 2004; revision accepted Aug. 8, 2004

Abstract: The paper investigates a semi on-line scheduling problem wherein the largest processing time of jobs done by three uniform machines M_1, M_2, M_3 is known in advance. A speed s_i (s_1=1, s_2=r, s_3=s, 1≤r≤s) is associated with machine M_i. Our goal is to maximize $C_{\min}$–the minimum workload of the three machines. We present a min3 algorithm and prove its competitive ratio is max{r+1,(3s+r+1)/(1+r+s)}, with the lower bound being at least max{2,r}. We also claim the competitive ratio of min3 algorithm cannot be improved and is the best possible for 1≤s≤2, r=1.

Key words: Scheduling, Semi on-line, Competitive ratio
doi:10.1631/jzus.2005.A0591 **Document code:** A **CLC number:** O223

INTRODUCTION

Sequencing and scheduling are forms of decision-making which play a crucial role in most manufacturing and production systems as well as in most information-processing environments, and also exist in transportation and distribution settings and in other types of service industries.

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 of the problem, such as total number of jobs to be scheduled, their release dates and their processing times, before solution algorithms are applied, such situation is termed off-line.

In many situations it is too optimistic to assume that we know everything about jobs as the off-line algorithm does, and too pessimistic to assume that we know nothing about jobs as the on-line algorithm does. A reasonable and practical assumption is that we know something about the jobs. This means that some partial information about jobs is available, such as the largest processing time being known in advance. If we cannot rearrange any job which has been assigned to machines, such a case is defined as a semi on-line problem.

The performance of an on-line or semi on-line algorithm is measured through the competitive ratio with respect to the optimal solution of its corresponding off-line problem. For a set J of jobs and an approximation algorithm A, if the objective is to maximize the $C_{\min}$ such as in our case, then the competitive ratio of algorithm A is defined as

$$R_A = \sup_J \left\{ \frac{C^*(J)}{C_A(J)} \right\},$$

where $C_A(J)$ denotes the value produced by the algorithm A and $C^*(J)$ denotes the optimal objective value in an off-line version.

The off-line version of the $P||C_{\min}$ which is intensively studied is known to be NP-complete in the strong sense (Garey and Johnson, 1979). For the on-line version of the $P||C_{\min}$, Woeginger (1997)

showed that LS algorithm is m competitive. It is well known that no deterministic algorithm can achieve a better competitive ratio. Deuermeyer *et al*.(1982) proved the competitive ratio of the classical LPT algorithm is at most 4/3. The tight ratio for this heuristic, $(4m-2)/(3m-1)$, is given by Csirik *et al*.(1992). Azar and Epstein (1997) showed that any randomized algorithm cannot have a competitive ratio less than $\sqrt{m}/4 = O(m)$ and presented a randomized algorithm called Partition with competitive ratio $O(\sqrt{m}\log m)$). He and Tan (2003) presented an optimal algorithm called Random which is 3/2 competitive on two machines, and derived a new randomized lower bound $\left[\left(\frac{m}{m-1}\right)^m - 1\right] \Big/ \left(\frac{m}{m-1}\right)$ for general m machines which significantly improves the known lower bound $\sqrt{m}/4$ for $m<41$.

For the semi on-line version of the $P||C_{\min}$, if the order of the jobs by their processing times is known in advance, He and Tan (2002) gave an ordinal algorithm called MIN algorithm which is at most $\left(\left\lfloor \sum_{i=1}^{m} 1/i \right\rfloor + 1\right)$−competitive for any m machine case, while the lower bound is $\sum_{i=1}^{m} 1/i$. Furthermore, for m=3, they presented an optimal algorithm called MIN(3) whose competitive ratio is 2. He (2001) got optimal deterministic algorithms for four semi on-line problems of $P2||C_{\min}$ where the total processing time of jobs is known in advance, the largest processing time of jobs is known in advance, the order of the jobs is known in advance, and the processing times of jobs are in the interval $[p, rp]$, where $p>0$ and $r\geq 1$, respectively. He (2000) further proved that LS is an optimal algorithm for the above last problem for any $m\geq 2$ and $r\geq 1$. For semi on-line version with non-increasing job processing times, He and Tan (2003) showed that LS algorithm is an optimal deterministic algorithm for two and three machine cases. They further presented an optimal randomized algorithm RLS for two machine case. For the case where the value of the optimal schedule is known in advance, Azar and Epstein (1997) proposed a deterministic algorithm called Fill, with competitive ratio $(2m-1)/m$, which is the best possible for m=2, 3 and 4. Epstein (2002) considered sequences where all jobs are bounded by OPT/k (for an integer k) on two identical machines and proved that GREEDY has optimal competitive ratio of $2k/(2k-1)$. For the case where OPT is known in advance Epstein showed that kFILL algorithm has optimal ratio of $(2k+1)/2k$.

For on-line or semi on-line version of the $Q||C_{\min}$ problem, Azar and Epstein (1997) gave several algorithms and lower bounds for m machines. The competitive ratios in their paper were given only as a function of m and not as a function of the speeds. However, for the case where the value of the optimal assignment is known in advance, and for the case where jobs arrive in non-increasing order, the tight competitive ratio is shown to be m. They further showed a constant 2 competitive algorithm for the intersection of the above two cases. Epstein (2002) showed that the exact competitive ratio is $1+q$ for two machines of speed ratio q. This competitive ratio grows linearly with q. To overcome the linearly growing competitive ratio Epstein studied the alternative model which restricts the input to sequences where OPT is known in advance. In this case Epstein provided a tight analysis of the competitive ratio function which is split into 3 intervals.

In this paper we consider a semi on-line version on three uniform machines M_1, M_2, M_3 (This semi on-line version on two uniform machines M_1, M_2 had been studied by Luo and Sun) where the processing time of the largest job is known in advance. This semi on-line version was first proposed by He and Zhang (1999) when $P2||C_{\max}$ was considered. A speed s_i ($s_1=1$, $s_2=r$, $s_3=s$, $1\leq r\leq s$) is associated with machine M_i. In one unit of time M_i can consume s_i units of processing time. A list $(p_1, p_2, \ldots, p_n)$ of jobs given to us on-line, means we get the jobs one by one but both the total number of jobs that must be scheduled and the size of the jobs are not previously known. The processing time of job p_i becomes known only when p_{i-1} has already been scheduled. As soon as job p_i appears it must irrevocably be scheduled. Our goal is to maximize the $C_{\min}$. We denote this semi on-line problem as $Q3|$known largest job$|C_{\min}$. We give a min3 algorithm and prove its competitive ratio is $\max\{r+1, (3s+r+1)/(1+r+s)\}$ which is the best possible value for $1\leq s\leq 2$ and $r=1$. We also claim the competitive ratio of min3 algorithm cannot be improved.

THE MIN3 ALGORITHM

Before we present our heuristics, we introduce some notations. Let $L(M_i)(i=1,2,3)$ denote machine M_i's workload, which is the proportion of the total processing times of all jobs that have been assigned to machine M_i with its speed. To simplify notation we use M_i instead of $L(M_i)$ if there is no confusion caused. We use $p_{\max}$ to denote the jobs' largest processing time. We call a job the largest job if its processing time on machine M_1 is $p_{\max}$. We also use $p_{\max}$ to denote the largest job and x to denote both the current job needed to be assigned to some machines at present and its processing time on machine M_1.

The min3 algorithm:

Step 1: For $x\neq p_{\max}$, if $\exists i\in\{1,2\}$ such that $M_i<p_{\max}/s$, assign the current job x to machine M_1 or M_2 which has the minimum workload, otherwise go to Step 3.

Step 2: For $x=p_{\max}$.

Step 2.1: If $M_3+x/s<2p_{\max}/s$, assign the current job x to machine M_3; otherwise go to Step 2.2.

Step 2.2: If $\exists i\in\{1,2\}$ such that $M_i<p_{\max}/s$, assign the current job x to machine M_1 or M_2 which has the minimum workload, otherwise go to Step 3.

Step 3: If $M_3+x/s\leq\min\{M_1+x,\ M_1+p_{\max}/s,\ M_2+p_{\max}/r,\ M_2+p_{\max}/s\}$, assign the current job x to machine M_3, otherwise assign the current job x to machine M_1 or M_2 by LS algorithm.

The heuristics terminate when no more job needs to be assigned.

THE UPPER AND LOWER BOUND

Before we present our main results we introduce two Lemmas which are useful in proving Theorem 1. Suppose $M_i\,(i=1,2,3)$ is machine M_i's workload at any time, then by min3 algorithm and LS algorithm it is easy to get Lemma 1.

Lemma 1 At any time,

1) $M_1-M_2\leq p_{\max}$, $M_2-M_1\leq p_{\max}/r$

2) $M_3-M_i\leq p_{\max}/s$ $(i=1,2)$.

Lemma 2 If $M_3\geq p_{\max}/s$, then $M_1-M_3\leq p_{\max}$, $M_2-M_3\leq p_{\max}/r$.

Proof Without loss of generality, we assume that at some time $M_3\geq p_{\max}/s$ but $M_1-M_3>p_{\max}$. Suppose job x is the last job assigned to machine M_1 and M_1^x is the workload of machine M_1 just before job x is assigned. So by $x\leq p_{\max}$, we have $M_1^x>M_3$. By min3 algorithm job x should not be assigned to machine M_1, this is in contradiction with the assumption. In the same way we have $M_2-M_3\leq p_{\max}/r$ and our proof is completed.

Theorem 1 The competitive ratio of the algorithm min3 for $Q3|\text{known largest job}|C_{\min}$ ($s_1=1$, $s_2=r$, $s_3=s$, $1\leq r\leq s$) is $\max\{r+1,(3s+r+1)/(1+r+s)\}$.

Proof In fact, we only need to prove $C^*(J)/C_{\min 3}(J)\leq\max\{r+1,(3s+r+1)/(1+r+s)\}$ holds for any instance J. Suppose that J is an instance and p_n is the last job. Immediately before p_n is assigned by min3, the workloads of three machines are denoted by M_1, M_2, M_3, respectively. We consider three cases as follows.

Case 1 $M_3<p_{\max}/s$.

In this case, it is easy to know that $p_n=p_{\max}$, $M_1<p_{\max}+p_{\max}/s$, $M_2<p_{\max}/r+p_{\max}/s$ and that job p_n should be assigned to machine M_3. We consider the ratio according to the following three subcases.

Subcase 1 $M_3+p_n/s\leq M_1$, $M_3+p_n/s\leq M_2$. It is obvious that

$$\frac{C^*(J)}{C_{\min 3}(J)}\leq\frac{M_1+rM_2+sM_3+p_n}{(1+r+s)(M_3+p_n/s)}<\frac{s}{1+r+s}+\frac{2p_{\max}+p_{\max}/s+rp_{\max}/s}{(1+r+s)p_{\max}/s}=\frac{3s+r+1}{1+r+s}.$$

Subcase 2 $M_2<M_3+p_n/s$, $M_2\leq M_1$.

If $M_2\geq p_{\max}/s$, then we have

$$\frac{C^*(J)}{C_{\min 3}(J)}\leq\frac{r}{1+r+s}+\frac{M_1+sM_3+p_n}{(1+r+s)M_2}<\frac{3s+r+1}{1+r+s}$$

If $M_2<p_{\max}/s$, we suppose x to be the last job assigned on machine M_1, $M_1^x=M_1-x$. It is easy to know that $M_1^x\leq M_2$, $M_3=0$. If $x\geq(1+r)M_2$, then $C^*(J)\leq(1+r)\times M_2$. It follows that $\frac{C^*(J)}{C_{\min 3}(J)}\leq\frac{(1+r)M_2}{M_2}=1+r$. If $x<(1+r)M_2$, then $C^*(J)<\frac{2(1+r)M_2}{1+r}=2M_2$. It implies that $\frac{C^*(J)}{C_{\min 3}(J)}\leq\frac{2M_2}{M_2}=2$.

Subcase 3 $M_1<M_3+p_n/s$, $M_1<M_2$. If $M_1\geq p_{\max}/s$, then we have

$$\frac{C^*(J)}{C_{\min 3}(J)} < \frac{1}{1+r+s} + \frac{3p_{\max} + rp_{\max}/s}{(1+r+s)p_{\max}/s} = \frac{3s+r+1}{1+r+s}$$

If $M_1<p_{\max}/s$, in the same way as $M_2<p_{\max}/s$ of Subcase 2 we have $\frac{C^*(J)}{C_{\min 3}(J)} \le r+1$.

Case 2 $M_3=p_{\max}/s$. We consider the following two subcases.

Subcase 1 If there is only one job on machine M_3, then it must be $p_{\max}$.

1) If job p_n is not assigned to machine M_3, then it corresponds to $M_3=0$ in Case 1.

2) If job p_n is assigned to machine M_3, then it is the end of executing Step 3, so we have $p_{\max}/s \le M_1 \le p_{\max}+p_{\max}/s$, $p_{\max}/s \le M_2 \le p_{\max}/s+p_{\max}/r$.

a) If $M_3+p_n/s \le M_2$, $M_3+p_n/s \le M_1$, then we have

$$\frac{C^*(J)}{C_{\min 3}(J)} < \frac{s}{1+r+s} + \frac{p_{\max} + \frac{p_{\max}}{s} + r\left(\frac{p_{\max}}{s} + \frac{p_{\max}}{r}\right)}{(1+r+s)\frac{p_{\max}}{s}} = \frac{3s+r+1}{1+r+s}.$$

b) $M_2<M_3+p_n/s$, $M_2 \le M_1$, then we have

$$\frac{C^*(J)}{C_{\min 3}(J)} \le \frac{r}{1+r+s} + \frac{3p_{\max} + \frac{p_{\max}}{s}}{(1+r+s)\frac{p_{\max}}{s}} = \frac{3s+r+1}{1+r+s}.$$

c) If $M_1<M_3+p_n/s$, $M_1 \le M_2$, then we have

$$\frac{C^*(J)}{C_{\min 3}(J)} \le \frac{1}{1+r+s} + \frac{3p_{\max} + \frac{rp_{\max}}{s}}{(1+r+s)\frac{p_{\max}}{s}} = \frac{3s+r+1}{1+r+s}.$$

Subcase 2 If there are at least two jobs on machine M_3, it is obvious that $p_{\max}/s \le M_1 \le p_{\max}+p_{\max}/s$, $p_{\max}/s \le M_2 \le p_{\max}/s+p_{\max}/r$ and job p_n should be assigned to machine M_3. Similarly as in 2) in Subcase 1 we can get the desired result.

Case 3 $M_3>p_{\max}/s$.

In this case, it is easy to know that $M_1 \ge p_{\max}/s$, $M_2 \ge p_{\max}/s$. We consider the ratio according to the following two subcases.

Subcase 1 Job p_n is assigned to machine M_1. Note that at present machine M_1's workload is M_1+p_n, by Lemma 2 we have $M_1+p_n \le M_3+p_{\max}$, $M_2 \le M_3+p_{\max}/r$.

1) $M_1+p_n \le M_2$, $M_1+p_n \le M_3$. By Lemma 1 we know $M_2 \le M_1+p_{\max}/r$, $M_3 \le M_1+p_{\max}/s$, then we have

$$\frac{C^*(J)}{C_{\min 3}(J)} \le \frac{1}{1+r+s} + \frac{(s+r)M_1 + 2p_{\max}}{(1+r+s)(M_1+p_n)} < 1 + \frac{2p_{\max}}{(1+r+s)p_{\max}/s} = \frac{3s+r+1}{1+r+s}.$$

2) $M_2<M_1+p_n$, $M_2 \le M_3$. By Lemma 1 we know $M_3 \le M_2+p_{\max}/s$, then we have

$$\frac{C^*(J)}{C_{\min 3}(J)} \le \frac{M_1 + rM_2 + sM_3 + p_n}{(1+r+s)M_2} \le 1 + \frac{2p_{\max}}{(1+r+s)p_{\max}/s} = \frac{3s+r+1}{1+r+s}.$$

3) $M_3<M_1+p_n$, $M_3<M_2$, then we have

$$\frac{C^*(J)}{C_{\min 3}(J)} \le \frac{M_1 + rM_2 + sM_3 + p_n}{(1+r+s)M_3} \le \frac{(s+r+1)M_3 + 2p_{\max}}{(1+r+s)M_3} < \frac{3s+r+1}{1+r+s}.$$

Subcase 2 If job p_n is assigned to machine M_2 or M_3, similarly we have the desired result.

Theorem 2 The competitive ratio of min3 algorithm cannot be smaller than $\max\{r+1,(3s+r+1)/(1+r+s)\}$.

Proof We consider two cases as follows.

Case 1 $r+1 \ge (3s+r+1)/(1+r+s)$.

Consider the following instance of four jobs $p_1=x$, $p_2=rx$, $p_3=s$, $p_4=r(1+r)x$. Suppose p_3 is the largest job and $x \le \frac{s}{r(1+r)}$, $(1+r)x \le 1$. By min3 algorithm, p_1 and p_4 should be assigned to machine M_1 and p_2, p_3 should be assigned to machine M_2, M_3, respectively. It is obvious that $C^*(J)=(1+r)x$ and that $C_{\min 3}(J)=x$ which follows $C^*(J)/C_{\min 3}(J)=r+1$.

Case 2 $r+1<(3s+r+1)/(1+r+s)$.

In this case, it is easy to know that $s>r(r+1)/(2-r)$. Suppose there is an instance of seven jobs:

$$p_1=1,\ p_2=r-\varepsilon,\ p_3=p_4=s,\ p_5=s-\frac{2(1+r)s}{s+r+1},$$
$$p_6=2-\frac{2(1+r)}{s+r+1},\ p_7=\left[2-\frac{2(1+r)}{s+r+1}\right]r,$$

where $p_{\max}=s$ and ε is an arbitrary small positive number. It is obvious that $C^*(J)=\dfrac{(3s+r+1)-\dfrac{\varepsilon}{r}(s+r+1)}{s+r+1}$. By min3 algorithm, it is easy to know $p_1\to M_1$, $p_2\to M_2$, $p_3\to M_3$, where $p_i\to M_i$ means that job p_i is assigned to machine M_i. Since the present workload of machine M_2 is $1-\varepsilon/r<p_{\max}/s=1$, by Step 2.2 of min3 algorithm, we know $p_4=s\to M_2$. By Step 3 of min3 algorithm, it is clear that all the rest of the jobs are assigned to machine M_3, therefore $C_{\min 3}=1$. Note that as ε is an arbitrary small positive number we have $R_{\min 3}=(3s+r+1)/(1+r+s)$.

Theorem 3 The competitive ratio of any algorithm for $Q3|\text{known largest job}|C_{\min}(s_1=1, s_2=r, s_3=s, 1\le r\le s)$ is at least $\max\{2,r\}$.

Proof We distinguish two cases according to the value of r.

Case 1 $r\ge 2$.

Suppose the processing time of the largest job is s and the first job is p_1 with a processing time rx ($r^2x\le p_{\max}$, $rx\le 1$). If algorithm A assigns p_1 to machine M_1, then $p_2=x$, $p_3=s$ come and no new job comes. It is obvious that $C^*(J)=x$ and $C_A\le x/r$. It follows that $C^*(J)/C_A(J)\ge r$. If algorithm A assigns it to machine M_2 or M_3, then $p_2=r^2x$, $p_3=s$ come and no new job comes. It is obvious that $C^*(J)=rx$ and $C_A\le x$. It implies that $C^*(J)/C_A(J)\ge r$.

Case 2 $1\le r<2$.

Suppose the processing time of the largest job is s and the first three jobs are $p_1=1/2$, $p_2=1/2$, $p_3=s$. In order to achieve a finite competitive ratio any algorithm should assigns them to different machines, respectively. We further assume $p_4=r$ and no new job comes. It is clear that $C^*(J)=1$ and $C_A(J)\le 1/2$. It follows that $C^*(J)/C_A(J)\ge 2$.

By Theorem 1 and Theorem 3, it is easy to have the following corollary.

Corollary 1 The competitive ratio of algorithm min3 for $Q3|\text{known largest job}|C_{\min}(s_1=s_2=1, 1\le s_3=s)$ is $\max\{2,(3s+2)/(2+s)\}$ which is the best possible value for $1\le s\le 2$.

CONCLUSION

In this paper, we present a min3 algorithm with competitive ratio of $\max\{r+1, (3s+r+1)/(1+r+s)\}$ and show the competitive ratio of min3 algorithm cannot be improved and is the best possible algorithm for $1\le s\le 2$, $r=1$. We further give a lower bound $\max\{2,r\}$. It is clear that the gap between the lower bound and upper bound will never exceed 1 and that min3 algorithm is more effective as r increases. It is interesting to notice that the competitive ratio of min3 algorithm is a function of r and not of s when $r\ge 2$. It is also interesting to extend this results to the general case $m\ge 4$.

References

Azar, Y., Epstein, L., 1997. On-line Machine Covering, Algorithms-ESA'97. Lecture Notes in Computer Science 1284, Springer Verlag, p.23-36.

Csirik, J., Kellerer, H., Woeginger, G., 1992. The exact LPT-bound for maximizing the minimum completion time. *Oper. Res. Letters*, **11**:281-287.

Deuermeyer, B.L., Friesen, D.K., Langston, M.A., 1982. Scheduling to maximize the minimum processor finish time in a mutiprocessor system. *SIAM J. Algorithms Discrete Methods*, **3**:190-196.

Epstein, L., 2002. Tight Bounds for Bandwidth Allocation on two Links. Proc. of the 3rd ARACNE, p.39-50.

Garey, M.R., Johnson, D.S., 1979. Computers and Intractability. W.H. Freeman and Company, San Francisco.

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

He, Y., 2001. Semi on-line scheduling problem for maximizing the minimum machine completion time. *Acta Mathematica Applicate Sinica*, **17**:107-113 (in Chinese).

He, Y., Zhang, G., 1999. Semi on-line scheduling on two identical machines. *Computing*, **62**(3):179-187.

He, Y., Tan, Z.Y., 2002. Ordinal on-line scheduling for maximizing the minimum machine completion time. *Journal of Combinatorial Optimization*, **6**:199-206.

He, Y., Tan, Z.Y., 2003. Randomized on-line and semi on-line scheduling on identical machines. *Asia-Pacific Journal of Operational Research*, **20**:31-40.

Woeginger, G., 1997. A polynomial time approximation scheme for minimizing the minimum machine completion time. *Oper. Res. Letters*, **20**:149-154.