Journal of the Operations Research
Society of Japan
2005, Vol. 48, No. 1, 1-8

© 2005 The Operations Research Society of Japan

SEMI-ON-LINE PARALLEL MACHINES SCHEDULING UNDER KNOWN TOTAL AND LARGEST PROCESSING TIMES

Soo Young Chang
Pohang University of Science and Technology

Hark-Chin Hwang
Chosun University

Jongho Park
Pohang University of Science and Technology

(Received September 4, 2003; Revised October 12, 2004)

Abstract We consider the semi-on-line parallel machines scheduling problem with the known total and the largest processing times of all jobs. We develop an approximation algorithm for the problem and prove that the competitive ratio of the algorithm is not more than $\sqrt{\frac{10}{3}} \approx 1.8257$, regardless of the number of machines.

Keywords: Scheduling, approximation algorithm, competitive analysis

1. Introduction

The scheduling problem is characterized as being *on-line* if the information for each job is not known in advance but revealed as it arrives and every job assignment must be made irrevocably in the order of arrival [3]. However, if partial information about the set of arriving jobs are known in advance, the problem is considered semi-on-line. In many online problems, one might know almost exact total and largest processing time in advance. For example, one may have good estimations for total processing time of all jobs in a certain period. Hence such estimations can be used as total processing time. Also in many cases the processing times of jobs are normally distributed and it is possible to give acceptable lower and upper bounds for the processing time of a job. So the upper bound can be employed as the largest processing time of a job. In this paper, we consider the semi-on-line parallel machines scheduling problem where the total and the largest processing times of all jobs are known in advance.

In order to evaluate the performance of algorithms for the semi-on-line scheduling problems, the competitive analysis is often conducted [9]. In the competitive analysis, we imagine that we have the makespan of the optimum *off-line* schedule which is obtainable given the complete information of all jobs in advance. Then, for a semi-on-line algorithm, say A, we derive its *competitive ratio* defined as the smallest value c satisfying $\frac{z^A(I)}{z^*(I)} \leq c$ for any instance I, where $z^A(I)$ and $z^*(I)$ denote the makespans of the schedule produced by A and the optimum off-line schedule for the problem instance I, respectively.

Variety of semi-on-line problems can be defined depending on the type of available a priori information. Kellerer et al. [7] defined the case of the semi-on-line problem where only the total processing time of all jobs is known in advance and presented an algorithm with competitive ratio of $\frac{4}{3}$ for the case of two machines. He and Zhang [5] studied the case of the known largest processing time of all jobs and presented an algorithm with competitive ratio $\frac{4}{3}$, for the case of two machines. Seiden et al. [8] analyzed the case where the jobs are known to arrive in non-increasing order of processing times and proved that the List Scheduling(LS for short) has its competitive ratio of $\frac{7}{6}$.

Nevertheless, for the cases of an arbitrary number of machines, we have results reported only for the *on-line* cases. Bartal et al. [2] proved that the competitive ratio of any algorithm for the on-line problem can never be less than 1.837 as the number of machines gets large enough. Albers [1] proposed an algorithm with its competitive ratio of 1.923 proven for all cases of an arbitrary number of machines. Therefore, it is certainly interesting to investigate the possibility of developing an algorithm with its competitive ratio smaller than 1.923 for the cases of an arbitrary number of machines.

In this paper, we consider the semi-on-line problem where both the total and largest processing time of all jobs are known. For this particular case, Tan and He [10] developed an algorithm with its competitive ratio $\frac{6}{5}$ which holds only for the problems with two machines. Hence, we propose a new algorithm with its competitive ratio proven to be not more than $\sqrt{\frac{10}{3}} \approx 1.8257$, regardless of the number of machines. In section 2, we introduce the scheme, *FF-SRT* (First Fit with Small job Reverse Trial), which is the key element of our proposed algorithm. The useful properties of *FF-SRT* are investigated in section 3 and based on these properties, we develop our algorithm, *LS-FF-SRT* (List Scheduling after *FF-SRT*) and establish an upper bound on its competitive ratio in section 4.

2. FF-SRT Scheme

We let $J = \{1, 2, \ldots, n\}$ be the set of job indices, arranged in the order of their arrivals. The jobs must be scheduled on m parallel identical machines irrevocably as they arrive. All machines are available at time zero and no preemption is allowed. The j^{th} job in J, or job j has its processing time denoted by p_j, which is not known until the job is about to be scheduled. However, we assume that the total and the largest processing time of all jobs, denoted respectively by sum and p_{max}, are known in advance.

We let S_i^j be the set of indices of jobs assigned to the i^{th} machine, or machine i, right after job j is scheduled, so that we use $S^j = \langle S_1^j, \ldots, S_m^j \rangle$ to denote the schedule that we have right after job j is scheduled. For the sake of notational completeness, we let S_i^0 be an empty set for $i = 1, \ldots, m$. We let $p(J')$ be the sum of processing time of all jobs in the set J', e.g., $p(S_i^j)$ is used to denote the total load on machine i right after job j is scheduled. Then the makespan, which is our objective to be minimized, of a schedule S^n may be defined as the maximum of $p(S_1^n)$ through $p(S_m^n)$. An instance of our problem is denoted by $I = (J, m; sum, p_{max})$, or I for short. Also, for our discussion, we denote the off-line optimum solution for I as $S^* = \langle S_1^*, \ldots, S_m^* \rangle$. Hence, $p(S_i^*) \leq z^*(I)$ for $1 \leq i \leq m$ and, therefore,

$$\textstyle\sum_{j \in J} p_j = \sum_{i=1}^{m} p(S_i^*) \leq m z^*(I). \tag{2.1}$$

In order to develop the scheme *FF-SRT* (*First Fit with Small job Reverse Trial*), we employ the well-known algorithm First Fit originally proposed for bin-packing problems [4], [6]. Given a pre-specified time deadline d, the First Fit assigns job j to machine i with the least index satisfying $p(S_i^{j-1}) + p_j \leq d$. Then, the First Fit for our problem suggests a Boolean function that we call FF as described below.

```
**Boolean** $FF(j, S^{j-1}, d)$
Step 1. $i = 1$.
Step 2. If $p(S_i^{j-1}) + p_j \leq d$, then assign job $j$ to machine $i$ and return *true*.
```

Step 3. If $i < m$, set $i = i + 1$ and go to step 2. Otherwise, return $false$.
Then, we define the Boolean function $RFF(ReverseFirst\text{-}Fit)$ which is essentially equivalent to FF, except that it assigns job j to machine i with the largest index satisfying $p(S_i^{j-1}) + p_j \le \frac{2}{5}d$.

```
**Boolean** $RFF(j, S^{j-1}, \frac{2}{5}d)$
Step 1. $i = m$.
Step 2. If $p(S_i^{j-1}) + p_j \le \frac{2}{5}d$, then assign job $j$ to machine $i$ and return $true$.
Step 3. If $i > 1$, set $i = i - 1$ and go to step 2. Otherwise, return $false$.
```

Then, we combine FF and RFF to develop the scheme $FF\text{-}SRT$ which is described as a Boolean function below.

```
**Boolean** $FF\text{-}SRT(j, S^{j-1}, d)$
Step 1. If $p_j \le \frac{1}{5}d$ and $RFF(j, S^{j-1}, \frac{2}{5}d)$ is $true$, then return $true$.
Step 2. Return $FF(j, S^{j-1}, d)$.
```

Note that the job with its processing time *small* relative to d is treated differently. In particular, we schedule every job with processing time not more than $\frac{1}{5}d$ using RFF while the time deadline is set as $\frac{2}{5}$ of d. If RFF returns $false$ for any small job, FF is invoked to schedule it. For the rest of the jobs, we simply use FF.

3. The Properties of FF-SRT

Note that $FF\text{-}SRT$ must return $true$ if d is sufficiently large. In this section, we prove the following theorem.

Theorem 3.1 $FF\text{-}SRT(j, S^{j-1}, d)$ always returns $true$ when $d \ge \frac{5}{3}z^*(I)$.

In order to facilitate the proof for this theorem, we classify jobs into three types with respect to the time deadline d; job j is *small* if $p_j \le \frac{1}{5}d$, *medium* if $\frac{1}{5}d < p_j \le \frac{2}{5}d$ and *big* if $\frac{2}{5}d < p_j$. To describe how many small jobs are contained in a job set $J' \subseteq J$, we use $\tilde{J}'$ to denote the set of small jobs, i.e., $\tilde{J}' = \{j : p_j \le \frac{1}{5}d, j \in J'\}$. Then, to make enumeration of medium and big jobs easy, we define the weight of each job j, denoted by w_j, as 0, $\frac{1}{2}$ and 1 if job j is small, medium and big, respectively. We also let $w(J')$ be the sum of the weights of all jobs in the set J', e.g., $w(S_i)$ is used to denote the total weight of jobs in S_i.

Then, if we choose $d \ge \frac{5}{3}z^*(I)$, each medium and big job has processing time strictly greater than $\frac{1}{3}$ and $\frac{2}{3}$ of $z^*(I)$, respectively. Hence, from $p(S_i^*) \le z^*(I)$ for all $i = 1, \ldots, m$, we must have

$$\textstyle\sum_{j \in J} w_j = \sum_{i=1}^{m} w(S_i^*) \le m. \tag{3.1}$$

Theorem 3.1 is going to be proved by contradiction. Hence, we suppose that Theorem 3.1 is false, so that there exists a counterexample, say I, such that $FF\text{-}SRT$ returns $false$ when $d \ge \frac{5}{3}z^*(I)$. Throughout the remainder of this section, we use I to denote a counterexample.

Supposing that a counterexample exists, we define job k to be the first job which caused $FF\text{-}SRT$ to return $false$ and S to be S^{k-1} which is the schedule that we have right before the job k is scheduled.

Property 3.1 Job k is big, i.e., $p_k > \frac{2}{5}d$ and $w_k = 1$.

Proof. If $p_k \leq \frac{2}{5}d$, $p(S_i) > \frac{3}{5}d$ for $1 \leq i \leq m$. But $d \geq \frac{5}{3}z^*(I)$ implies $p(S_i) > z^*(I)$ for $1 \leq i \leq m$, which contradicts (2.1). ♠

Property 3.2 $p(S_i) > \frac{2}{5}d$ for all $1 \leq i \leq m$ and S_i must contain at least one medium or big job if no small job is assigned to machine i by FF.

Proof. Clearly, no job has processing time greater than $z^*(I)$, implying $p_k \leq \frac{3}{5}d$ when $d \geq \frac{5}{3}z^*(I)$. Hence, $p(S_i) > \frac{2}{5}d$ for all $1 \leq i \leq m$. Hence, if no small job is assigned to machine i by FF, $p(S_i) > \frac{2}{5}d$ implies that S_i must contain at least one medium or big job, since all small jobs scheduled by RFF must be completed no later than $\frac{2}{5}d$. ♠

We also classify machines to facilitate our discussion. In particular, we classify machines into two types; machine i is classified as being *forward* and *reverse* if its first job is assigned by FF and RFF, respectively. Recall that FF may handle small jobs but only when RFF returns $false$. RFF never returns $false$, however, when there is an empty machine. Hence, the first job of any forward machine can never be a small job, whereas the first job of a reverse machine is always a small job. If machine f becomes forward when job $j < k$ is assigned to it by FF, S_f^{j-1} must be empty. From this, we know that job j is medium or big and RFF has never returned $false$ until the job j is scheduled. Moreover, we see that each S_i^j does not contain any small job and thus each machine i is also forward for $1 \leq i \leq f$. But, the very fact that job j is assigned to machine f implies that $p(S_i^j) > \frac{2}{5}d$ for $1 \leq i < f$. Therefore, each S_i^j contains at least one big job or at least two medium jobs for $1 \leq i < f$. Hence, we have $w(S_i^j) \geq 1$ and thus $w(S_i) \geq 1$. For the machine f, since job j is medium or big, it holds $w(S_f^j) \geq \frac{1}{2}$ and thus $w(S_f) \geq \frac{1}{2}$. We summarize the result formally in the following.

Property 3.3 Suppose that machine f is forward for some $1 \leq f \leq m$. Then, we have (a) each machine i is also forward with $w(S_i) \geq 1$ for $1 \leq i < f$, and (b) $w(S_f) \geq \frac{1}{2}$.

Now, consider the property of reverse machines. If a small job $j < k$ is assigned by RFF as the first job of machine r, then for all $r+1 \leq i \leq m$, each machine i is also reverse and the total sum of processing times of small jobs in S_i^{j-1} must be greater than $\frac{1}{5}d$, since the job j was failed to be put on the machines m to $r+1$ within time $\frac{2}{5}d$ and $p_j \leq \frac{1}{5}d$. Thus, $p(\tilde{S}_i^{j-1}) > \frac{1}{5}d$ holds, which means $p(\tilde{S}_i) > \frac{1}{5}d$. Hence, we have the next property.

Property 3.4 If machine r is reverse, then each machine i is also reverse and $p(\tilde{S}_i) > \frac{1}{5}d$ for $r+1 \leq i \leq m$.

The existence of reverse machines are dealt with in the next property.

Property 3.5 There exists at least one reverse machine.

Proof. Suppose that all the machines are forward. Then, by Property 3.3, we know that

$$\sum_{i=1}^{m-1} w(S_i) \geq m-1 \text{ and } w(S_m) \geq \tfrac{1}{2}.$$

Then, from this and the fact that $w_k = 1$ by Property 3.1, we have a contradiction to (3.1). ♠

The assignment of medium jobs plays a crucial role in our analysis. Regarding the existence of a medium job, we establish the following property.

Property 3.6 At least one medium job must be scheduled before job k.

Proof. The proof is by contradiction. So, we suppose that this property is false and no medium job has arrived before job k. Under this presupposition, we consider two possible cases, namely, the case where RFF never returned $false$ and the case where RFF returned $false$ at least once.

First, consider the case where RFF never returned $false$. Then no small job can be handled by FF. And, by Property 3.2, S_i must contain at least one big job, since we supposed that there is no medium job. Hence, $w(S_i) \geq 1$ for $i = 1, \ldots, m$. But since $w_k = 1$, a contradiction to (3.1) results.

Next, consider the case where RFF returned *false* at least once. In this case, at least one small job must have been scheduled by FF. Among such small jobs handled by FF, let job $j < k$ be the one assigned to the machine with the largest index, say t. Then, since we suppose that no medium job has arrived before job k, Property 3.2 implies that each of S_{t+1} through S_m must contain at least one big job. Hence,

$$S_i \text{ contains at least one big job and } w(S_i) \geq 1 \text{ for } t+1 \leq i \leq m. \tag{3.2}$$

Then, we consider two subcases where machine t is forward and reverse, respectively. If machine t is forward, then $w(S_t) \geq \frac{1}{2}$ and $w(S_i) \geq 1$ for $1 \leq i < t$ by Property 3.3. This together with (3.2) and $w_k = 1$ leads to a contradiction to (3.1).

Finally, consider the case where machine t is reverse. In this case, from the facts that job j is assigned to machine t by FF and $p_j \leq \frac{1}{5}d$, we note

$$p(S_i) > \tfrac{4}{5}d > z^*(I), \text{ for } i = 1, \ldots, t-1. \tag{3.3}$$

Also, by Property 3.4 and (3.2), we have

$$p(S_i) = p(\tilde{S}_i) + p(S_i - \tilde{S}_i) > \tfrac{1}{5}d + \tfrac{2}{5}d \geq z^*(I), \text{ for } i = t+1, \ldots, m. \tag{3.4}$$

Then, since $p(S_t) + p_k > d > z^*(I)$, this with (3.3) and (3.4) contradicts (2.1). ♠

By Property 3.5 and Property 3.6, we know that there exists at least one reverse machine and at least one medium job. So, from now on, we let u be the machine with the largest index among the machines with medium jobs and v be the machine with the smallest index among all reverse machines. Then, with this definition, we show Theorem 3.1.

Proof of Theorem 3.1 Suppose Theorem 3.1 does not hold. Then, all the properties developed so far hold. We consider two cases.

Case 1. $u \geq v$. Since a medium job is assigned to machine u instead of other machines 1 through $u-1$, we observe that

$$p(S_i) > \tfrac{3}{5}d \geq z^*(I), \text{ for } i = 1, \ldots, u-1. \tag{3.5}$$

Then, we further divide this case into two subcases, i.e., when RFF never returned $false$ and when RFF returned $false$ at least once. If RFF never returned $false$, no small job can be scheduled by FF. Then, by Property 3.2 with the definition of u, S_i must contain at least one big job but no medium job for $i = u+1, \ldots, m$. But, since $u \geq v$, each machine i is reverse for $i = u+1, \ldots, m$. Then by Property 3.4, we have

$$p(S_i) = p(\tilde{S}_i) + p(S_i - \tilde{S}_i) > \tfrac{1}{5}d + \tfrac{2}{5}d \geq z^*(I) \text{ for } i = u+1, \ldots, m. \tag{3.6}$$

Then, (3.5) and (3.6) together with the fact that $p(S_u) + p_k > d > z^*(I)$ contradict (2.1).

Next, we consider the subcase where RFF returned $false$ at least once. Then, there must exist at least one small job scheduled by FF. Among the machines to which a small job is assigned by FF, we let machine t be the one with the largest index and job j be a small job assigned to machine t by FF. If $t \leq u$, however, (3.6) must still hold by the definition of t and, therefore, we get a contradiction to (2.1). For the case where $t > u$, noting that a small job j is assigned to machine t instead of machines 1 through $t-1$, we obtain

$$p(S_i) > \tfrac{4}{5}d > z^*(I), \text{ for } i = 1, \ldots, t-1. \tag{3.7}$$

Since $t > u \geq v$, each machine i for $i = t+1, \ldots, m$ is reverse. But no medium job can be scheduled to any machine i for $i = u+1, \ldots, m$ due to the definition of u. Hence, by Property 3.2 and the definition of t, for $i = t+1, \ldots, m$, S_i must contain at least one big job. Thus, we get by Property 3.4

$$p(S_i) = p(\tilde{S}_i) + p(S_i - \tilde{S}_i) > \tfrac{1}{5}d + \tfrac{2}{5}d \geq z^*(I), \text{ for } i = t+1, \ldots, m. \tag{3.8}$$

Hence, by $p(S_t) + p_k > d > z^*(I)$ together with (3.7) and (3.8), we get a contradiction to (2.1).

Case 2. $u < v$. Since $u < v$, machine u is forward. Then, by Property 3.3, we get

$$\textstyle\sum_{i=1}^{u-1} w(S_i) \geq u-1 \text{ and } w(S_u) \geq 1/2. \tag{3.9}$$

At this point, we further divide this case into two subcases, i.e., the case when RFF never returned $false$ and the case otherwise. First, if RFF never returned $false$, no small job is scheduled by FF. By Property 3.2 and the definition of u, machine i for $i = u+1, \ldots, m$ must have at least one big job. Hence, we get

$$\sum_{i=u+1}^{m} w(S_i) \geq m - u. \tag{3.10}$$

Since $w_k = 1$, from (3.9) and (3.10), we have a contradiction to (3.1). Now, suppose that RFF returned $false$ at least once before job k, implying that there exists at least one small job assigned by FF. Among such small jobs, let job j be the small one assigned by FF to the machine with the largest index, say t. First, we note that $u < v$ implies $v > 1$ and machine v-1 is forward. Then, by Property 3.3,

$$\textstyle\sum_{i=1}^{v-2} w(S_i) \geq v-2 \text{ and } w(S_{v-1}) \geq 1/2. \tag{3.11}$$

If $t < v$, by Property 3.2, each S_i must contain at least one big job for $i = v, \ldots, m$, since $u < v$ in this case. So, we have

$$\sum_{i=v}^{m} w(S_i) \geq m - v + 1. \tag{3.12}$$

Since $w_k = 1$, (3.11) and (3.12) lead to a contradiction to (3.1).

Finally, consider when $t \geq v$. Then the fact that the small job j is assigned to machine t implies,

$$p(S_i) > \tfrac{4}{5}d > z^*(I), \text{ for } i = 1, \ldots, t-1. \tag{3.13}$$

$t \geq v$ implies that machine t is reverse. Then, by the definition of t and u along with Property 3.2, we know that each S_i for $i = t+1, \ldots, m$ must contain at least one big job. Hence, by Property 3.4, we have

$$p(S_i) > \tfrac{3}{5}d \geq z^*(I), \text{ for } i = t+1, \ldots, m. \tag{3.14}$$

Since $p(S_t) + p_k > d > z^*(I)$, from (3.13) and (3.14), we get a contradiction to (2.1).

4. Algorithm LS-FF-SRT

We propose our algorithm *LS-FF-SRT* as described below.

```
Algorithm $LS\text{-}FF\text{-}SRT(j, S^{j-1}, d)$
Step 1. If $FF\text{-}SRT(j, S^{j-1}, d)=true$, end.
Step 2. Assign job $j$ to machine $i$ with the smallest $p(S_i^{j-1})$ (List Scheduling).
```

We now conclude our paper, establishing upper bound on the competitive ratio of the proposed algorithm with its time deadline d set as

$$d = \sqrt{\tfrac{10}{3}}\pi \text{ where } \pi = \max\{\tfrac{sum}{m}, p_{max}\}.$$

Theorem 4.1 *The competitive ratio of LS-FF-SRT with* $d = \sqrt{\frac{10}{3}}\pi$ *is not more than* $\sqrt{\frac{10}{3}}$.

Proof. We know that $\sqrt{\frac{10}{3}}z^*(I) \geq \sqrt{\frac{10}{3}}\pi$ because $z^*(I) \geq \max\{\frac{sum}{m}, p_{max}\}$. Hence, it is enough to show that *LS-FF-SRT* always yields a schedule with its makespan less than or equal to $\sqrt{\frac{10}{3}}z^*(I)$ in order to prove the theorem.

If $\sqrt{\frac{10}{3}}\pi \geq \frac{5}{3}z^*(I)$, the theorem follows from Theorem 1 proved in the previous section. Hence, we prove the theorem for the case where $\sqrt{\frac{10}{3}}\pi < \frac{5}{3}z^*(I)$. To this end, we suppose that the theorem does not hold and, so, there exists a counterexample I such that the makespan of the schedule generated by *LS-FF-SRT* is more than $\sqrt{\frac{10}{3}}z^*(I)$. We let job k be the first job scheduled to be completed after $\sqrt{\frac{10}{3}}z^*(I)$ and $S = S^{k-1}$. Then, if we let $\omega = \frac{z^*(I)}{\pi}$, we have

$$p_k + p(S_i) > \sqrt{\tfrac{10}{3}}\omega\pi \text{ for } i = 1, \ldots, m. \tag{4.1}$$

By definition, we know $m\pi \geq sum > \sum_{i=1}^{m} p(S_i)$ implying that there exists machine i such that $p(S_i) < \pi$. Hence, from (4.1), we get $p_k > (\sqrt{\frac{10}{3}}\omega - 1)\pi$. Note that $\omega > \sqrt{\frac{6}{5}}$ when $\sqrt{\frac{10}{3}}\pi < \frac{5}{3}z^*(I)$ and therefore, $\sqrt{\frac{10}{3}}\omega > 2$. Hence,

$$p_k > (\sqrt{\tfrac{10}{3}}\omega - 1)\pi > \pi,$$

which is a clear contradiction to the definition of π. ♠

We conclude our paper with the following example showing that our competitive ratio is tight.

Example 1. There are $m+1$ jobs to be scheduled on m machines. The total processing time (sum) and the largest processing time (p_{max}) are known as m and 1, respectively. Thus, for this instance I, we have $\pi = \max\{\frac{sum}{m}, p_{max}\} = 1$ and the time deadline $d = \sqrt{\frac{10}{3}}\pi = \sqrt{\frac{10}{3}}$. Each of the first $m-1$ jobs has the same processing time of 1 and the jobs m and $m+1$ have 0.8257 and 0.1743, respectively. Jobs are arriving in the sequence of $1, 2 \ldots, m$ and $m+1$. We first consider the *LS-FF-SRT* schedule. Since $p_i > \frac{1}{5}d \approx 0.3651$ for each job $i = 1, 2, \ldots, m-1$, *FF* of *LS-FF-SRT* is invoked to schedule the first $m-1$ jobs. Each of these jobs is separately assigned to machine i for $i = 1, 2, \ldots, m-1$ because any two such jobs cannot be completed within the time deadline $d \approx 1.8257$. Then job m with processing time of 0.8257 arrives. Since $p_m > 0.3651$, *FF* is also invoked to schedule it to machine 1 because $p_m + p(S_1^{m-1}) \leq 1.8257$. Finally job $m+1$ with processing time of 0.1743 arrives. Since $p_{m+1} \leq 0.3651$, RFF of *LS-FF-SRT* is invoked to schedule it to machine m. Thus, the makespan of the final schedule is $z^{LS-FF-SRT}(I) = p(S_1^{m+1}) \approx 1.8257$. In optimal off-line schedule of I, job m and job $m-1$ are scheduled to the same machine and the other jobs with processing time of 1 are assigned to different $m-1$ machines one by one. Hence we have $z^*(I) = 1$. Therefore, considering the makespan of the *LS-FF-SRT* schedule, we conclude that the ratio of 1.8257 is tight.

Acknowledgement

Hark-Chin Hwang was supported in part by research grant of Chosun University, 2005.

References

[1] S. Albers: Better bounds for online scheduling. *Proceeding of Twenty-Ninth Annual ACM Symposium on Theory of Computing*, (1997), 130-139.

[2] Y. Bartal, A. Fiat and Y. Rabani: A better lower bound for on-line scheduling. *Information Processing Letters*, **50** (1994), 113-116.

[3] B. Chen, C. N. Potts and G. Woeginger: A review of machine scheduling: Complexity, algorithms and approximability. In D.-Z. Du and P. M. Pardalos (eds.): Handbook of Combinatorial Optimization (Kluwer Academic Publishers, Dordrecht, 1998).

[4] M. R. Garey and D. S. Johnson: *Computers and Intractability: A Guide to the Theory of NP-Completeness* (Freeman, San Francisco, 1979).

[5] Y. He and G. C. Zhang: Semi-on-line scheduling on two identical machines. *Computing*, **62** (1999), 179-187.

[6] D. S. Johnson, A. Demers, J. D. Ullman, M. R. Garey and R. L. Graham: Worst-case performance bounds for simple one-dimensional packing algorithms. *SIAM Journal on Computing*, **3-4** (1974), 299-325.

[7] H. Kellerer, V. Kotov, M. G. Speranza and Z. Tuza: Semi on-line algorithms for the partition problem. *Operations Research Letters*, **21** (1997), 235-242.

[8] S. Seiden, J. Sgall and G. Woeginger: Semi-online scheduling with decreasing job sizes. *Operations Research Letters*, **27** (2000), 215-227.

[9] D. Sleator and R. E. Tarjan: Amortized efficiency of list update and paging rules. *Communications of the ACM*, **28** (1985), 202-208.

[10] Z. Tan and Y. He: Semi-on-line problems on two identical machines with combined partial information. *Operations Research Letters*, **30** (2002), 404-414.

Hark-Chin Hwang
Department of Industrial Engineering
Chosun University
375 Seosuk-Dong, Dong-Gu
Gwangju 501-709, South Korea
E-mail: `hchwang@chosun.ac.kr`