6

On-line Scheduling on Identical Machines for Jobs with Arbitrary Release Times

Rongheng Li[1] and Huei-Chuen Huang[2]
[1]*Department of Mathematics, Hunan Normal University,*
[2]*Department of Industrial and Systems Engineering, National University of Singapore*
[1]*China.*, [2]*Singapore*

1. Introduction

In the theory of scheduling, a problem type is categorized by its machine environment, job characteristic and objective function. According to the way information on job characteristic being released to the scheduler, scheduling models can be classified in two categories. One is termed off-line in which the scheduler has full information of the problem instance, such as the total number of jobs to be scheduled, their release times and processing times, before scheduling decisions need to be made. The other is called on-line in which the scheduler acquires information about jobs piece by piece and has to make a decision upon a request without information of all the possible future jobs. For the later, it can be further classified into two paradigms.

1. Scheduling jobs over the job list (or one by one). The jobs are given one by one according to a list. The scheduler gets to know a new job only after all earlier jobs have been scheduled.
2. Scheduling jobs over the machines' processing time. All jobs are given at their release times. The jobs are scheduled with the passage of time. At any point of the machines' processing time, the scheduler can decide whether any of the arrived jobs is to be assigned, but the scheduler has information only on the jobs that have arrived and has no clue on whether any more jobs will arrive.

Most of the scheduling problems aim to minimize some sort of objectives. A common objective is to minimize the overall completion time C_{max}, called makespan. In this chapter we also adopt the same objective and our problem paradigm is to schedule jobs on-line over a job list. We assume that there are a number of identical machines available and measure the performance of an algorithm by the worst case performance ratio. An on-line algorithm is said to have a worst case performance ratio σ if the objective of a schedule produced by the algorithm is at most σ times larger than the objective of an optimal off-line algorithm for any input instance.

For scheduling on-line over a job list, Graham (1969) gave an algorithm called List Scheduling (LS) which assigns the current job to the least loaded machine and showed that LS has a worst case performance ratio of $2-\frac{1}{m}$ where m denotes the number of machines available. Since then no better algorithm than LS had been proposed until Galambos & Woeginger (1993) and Chen et al. (1994) provided algorithms with better performance for

Source: Multiprocessor Scheduling: Theory and Applications, Book edited by Eugene Levner,
ISBN 978-3-902613-02-8, pp.436, December 2007, Itech Education and Publishing, Vienna, Austria

$m \geq 4$. Essentially their approach is to schedule the current job to one of the two least loaded machines while maintaining some machines lightly loaded in anticipation of the possible arrival of a long job. However for large m, their performance ratios still approach 2 because the algorithms leave at most one machine lightly loaded. The first successful approach to bring down the ratio from 2 was given by Bartal et al. (1995), which keeps a constant fraction of machines lightly loaded. Since then a few other algorithms which are better than LS have been proposed (Karger et al. 1996, Albers 1999). As far as we know, the current best performance ratio is 1.9201 which was given by Fleischer & Wahl (2000).

For scheduling on-line over the machines' processing time, Shmoys et al. (1995) designed a non-clairvoyant scheduling algorithm in which it is assumed any job's processing time is not known until it is completed. They proved that the algorithm has a performance ratio of 2. Some other work on the non-clairvoyant algorithm was done by Motwani et al. (1994). On the other hand, Chen & Vestjens (1997) considered the model in which jobs arrive over time and the processing time is known when a job arrives. They showed that a dynamic LPT algorithm, which schedules an available job with the largest processing time once a machine becomes available, has a performance ratio of 3/2.

In the literature, when job's release time is considered, it is normally assumed that a job arrives before the scheduler needs to make an assignment on the job. In other words, the release time list synchronizes with the job list. However in a number of business operations, a reservation is often required for a machine and a time slot before a job is released. Hence the scheduler needs to respond to the request whenever a reservation order is placed. In this case, the scheduler is informed of the job's arrival and processing time and the job's request is made in form of order before its actual release or arrival time. Such a problem was first proposed by Li & Huang (2004), where it is assumed that the orders appear on-line and upon request of an order the scheduler must irrevocably pre-assign a machine and a time slot for the job and the scheduler has no clue or whatsoever of other possible future orders. This problem is referred to as an on-line job scheduling with arbitrary release times, which is the subject of study in the chapter. The problem can be formally defined as follows. For a business operation, customers place job orders one by one and specify the release time r_j and the processing time p_j of the requested job J_j. Upon request of a customer's job order, the operation scheduler has to respond immediately to assign a machine out of the m available identical machines and a time slot on the chosen machine to process the job without interruption. This problem can be viewed as a generalization of the Graham's classical on-line scheduling problem as the later assumes that all jobs' release times are zero.

In the classical on-line algorithm, it is assumed that the scheduler has no information on the future jobs. Under this situation, it is well known that no algorithm has a better performance ratio than LS for $m \leq 3$ (Faigle et al. 1989) . It is then interesting to investigate whether the performance can be improved with additional information. To respond to this question, the semi-online scheduling is proposed. In the semi-online version, the conditions to be considered online are partially relaxed or additional information about jobs is known in advance and one wishes to make improvement of the performance of the optimal algorithm with respect to the classical online version. Different ways of relaxing the conditions give rise to different semi-online versions (Kellerer et al. 1997). Similarly several types of additional information are proposed to get algorithms with better performance. Examples include the total length of all jobs is known in advance (Seiden et al. 2000), the largest length of jobs is known in advance (Keller 1991, He et al. 2007), the lengths of all jobs are known in

[*p, rp*] where $p > 0$ and $r \geq 1$ which is called on-line scheduling for jobs with similar lengths (He & Zhang 1999, Kellerer 1991), and jobs arrive in the non- increasing order of their lengths (Liu et al. 1996, He & Tan 2001, 2002, Seiden et al. 2000). More recent publications on the semi-online scheduling can be found in Dosa et al. (2004) and Tan & He (2001, 2002). In the last section of this chapter we also extend our problem to be semi-online where jobs are assumed to have similar lengths.

The rest of the chapter is organized as follows. Section 2 defines a few basic terms and the LS algorithm for our problem. Section 3 gives the worst case performance ratio of the LS algorithm. Section 4 presents two better algorithms, MLS and NMLS, for $m \geq 2$. Section 5 proves that NMLS has a worst case performance ratio not more than 2.78436. Section 6 extends the problem to be semi-online by assuming that jobs have similar lengths. For simplicity of presentation, the job lengths are assumed to be in [l, *r*] or *p* is assumed to be 1. In this section the LS algorithm is studied. For $m \geq 2$, it gives an upper bound for the performance ratio and shows that 2 is an upper bound when $r \leq \frac{m}{m-1}$. For $m = 1$, it shows that the worst case performance ratio is $1 + \frac{r}{1+r}$ and in addition it gives a lower bound for the performance ratio of any algorithm.

2. Definitions and algorithm LS

Definition 1. Let $L = \{J_1, J_2, \ldots, J_n\}$ be any list of jobs, where job $J_j (j = 1, 2, \ldots, n)$ arrives at its release time r_j and has a processing time of p_j. There are m identical machines available. Algorithm A is a heuristic algorithm. $C_{max}^{A}(L)$ and $C_{max}^{OPT}(L)$ denote the makespans of algorithm A and an optimal off-line algorithm respectively. The worst case performance ratio of Algorithm A is defined as

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

Definition 2. Suppose that J_j is the current job with release time r_j and processing time p_j. Machine M_i is said to have 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 idle in interval $[T_1, T_2]$ and a job has been assigned on M_i to start processing at time T_2.

2. $T_2 - max\{T_1, r_j\} \geq p_j$.

It is obvious that if machine M_i has an idle time interval $[T_1, T_2]$ for job J_j, then job J_j can be assigned to machine M_i in the idle interval.

Algorithm LS

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

Step 2. If there exist some machines which have idle intervals for job J_n, then select a machine M_i which has an idle interval $[T_1, T_2]$ for job J_n with minimal T_1 and assign job J_n to machine M_i to be processed starting at time $\max\{T_1, r_n\}$ in the idle interval. Otherwise go to Step 3.

Step 3. Let $s = \max\{r_n, L_1\}$. Job J_n is assigned to machine M_1 at time s to start the processing.

We say that a job sequence $J_{i_1}, J_{i_2}, \cdots, J_{i_q}$ is assigned on machine M_i if J_{i_1} starts at its release time and $J_{i_k}(k = 2, \cdots, q)$ starts at its release time or the completion time of $J_{i_{k-1}}$, depending on which one is bigger.
In the following we let $M_i^{LS} = (J_{i_1}, J_{i_2}, \cdots, J_{i_q})$ denote the job list assigned on machine M_i in the *LS* schedule and $M_i^{OPT} = (J_{i_1}, J_{i_2}, \cdots, J_{i_q})$ denote the job list assigned on machine M_i in an optimal off-line schedule, where $J_{i_s} \in \{J_1, J_2, \cdots, J_n\}$ (s = 1, 2, ... , q).

3. Worst case performance of algorithm LS

For any job list $L = \{J_1, J_2, \ldots, J_n\}$, if $r_1 \leq r_2 \leq \ldots \leq r_n$, it is shown that $R(m, LS) \leq 2$ in Hall and Shmoys(1989). In the next theorem we provide the exact performance ratio.
Theorem 1 For any job list $L = \{J_1, J_2, \ldots, J_n\}$, if $r_1 \leq r_2 \leq \ldots \leq r_n$, then we have

$$R(m, LS) = 2 - \frac{1}{m}. \tag{1}$$

Proof: We will prove this theorem by argument of contradiction. Suppose that there exists an instance L, called a counterexample, satisfying:

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

Let $L = \{J_1, J_2, \ldots, J_n\}$ be a minimal counterexample, i.e., a counterexample consisting of a minimum number of jobs. It is easy to show that, for a minimal counterexample L, $C_{max}^{LS}(L) = L_1 + p_n$ holds.
Without loss of generality we can standardize L such that $r_1 = 0$. Because if this does not hold, we can alter the problem instance by decreasing the releasing times of all jobs by r_1. After the altering, the makespans of both the LS schedule and the optimal schedule will decrease by r_1, and correspondingly the ratio of the makespans will increase. Hence the altered instance provides a minimal counterexample with $r_1 = 0$.
Next we show that, at any time point from 0 to $C_{max}^{LS}(L)$, at least one machine is not idle in the LS schedule. If this is not true, then there is a common idle period within time interval $[0, C_{max}^{LS}(L)]$ in the LS schedule. Note that, according to the LS rules and the assumption that $r_1 \leq r_2 \leq \ldots \leq r_n$, jobs assigned after the common idle period must be released after this period. If we remove all the jobs that finish before this idle period, then the makespan of the LS schedule remains the same as before, whereas the corresponding optimal makespan does not increase. Hence the new instance is a smaller counterexample, contradicting the minimality. Therefore we may assume that at any time point from 0 to $C_{max}^{LS}(L)$ at least one machine is busy in the LS schedule.
As $r_1 \leq r_2 \leq \ldots \leq r_n$, it is also not difficult to see that no job is scheduled in Step 2 in the LS schedule.
Now we consider the performance ratio according to the following two cases:
Case 1. The LS schedule of L contains no idle time.
In this case we have

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

Case 2. There exists at least a time interval during which a machine is idle in the *LS* schedule. In this case, let [a, *b*] be such an idle time interval with $a < b$ and b being the biggest end point among all of the idle time intervals. Set

$$A = \{J_j \mid J_j \text{ finishes after time } b \text{ in the LS schedule}\}.$$

Let B be the subset of A consisting of jobs that start at or before time a. Let $S(J_j)(j = 1, 2, \ldots, n)$ denote the start time of job J_j in the LS schedule. Then set B can be expressed as follows:

$$B = \{J_j \mid b - p_j < S(J_j) \leq a\}.$$

By the definitions of A and B we have $S(J_j) > a$ for any job $J_j \in A \setminus B$. If both $r_j < b$ and $r_j < S(J_j)$ hold for some $J_j \in A \setminus B$, we will deduce a contradiction as follows. Let $L_i^{(j)}(i = 1, 2, \cdots, m)$ be the completion times of M_i just before job J_j is assigned in the LS schedule. First observe that during the period [a, *b*], at least one machine must be free in the LS schedule. Denote such a free machine by M_{i_1} and let M_{i_2} be the machine to which J_j is assigned. Then $a < S(J_j) = L_{i_2}^{(j)}$ because $r_j < S(J_j)$ and J_j is assigned by Step 3. On the other hand we have that $L_{i_1}^{(j)} \leq a$ because $r_j < b$ and M_{i_1} must be free in (a, ∞) in the *LS* schedule before J_j is assigned as all jobs assigned on machine M_{i_1} to start at or after b must have higher indices than job J_j. This implies $L_{i_1}^{(j)} < L_{i_2}^{(j)}$ and job J_j should be assigned to machine M_{i_1}, contradicting the assumption that job J_j is assigned to machine M_{i_2} instead. Hence, for any job $J_j \in A \setminus B$, either $r_j \geq b$ or $r_j = S(J_j)$. As a consequence, for any job $J_j \in A \setminus B$, the processing that is executed after time b in the LS schedule cannot be scheduled earlier than b in any optimal schedule. Let $\Delta = 0$ if B is empty and $\Delta = \max\{S(J_j) - r_j | J_j \in B\}$ if B is not empty. It is easy to see that the amount of processing currently executed after b in the LS schedule that could be executed before b in any other schedule is at most $\Delta|B|$. Therefore, taking into account that all machines are busy during $[b, L_1]$ and that $|B| \leq m - 1$, we obtain the following lower bound based on the amount of processing that has to be executed after time b in any schedule:

$$\begin{aligned}C_{max}^{OPT}(L) &\geq b + \frac{m(L_1 - b) + p_n - |B|\Delta}{m} \\ &\geq b + \frac{m(L_1 - b) + p_n - (m-1)\Delta}{m}.\end{aligned}$$

On the other hand let us consider all the jobs. Note that, if $\Delta > 0$, then in the LS schedule, there exists a job J_j with $S(J_j) \leq a$ and $S(J_j) - r_j = \Delta$. It can be seen that interval $[r_j, S(J_j)]$ is a

period of time before a with length of Δ, during which all machines are busy just before J_j is assigned. This is because no job has a release time bigger than r_j before J_j is assigned and, by the facts that $S(J_j) - r_j = \Delta > 0$, and $S(J_j) = \min\{L_i^{(j)} | i = 1, 2, \cdots, m\}$. Combining with the observations that during the time interval $[b, L_1]$ all machines are occupied and at any other time point at least one machine is busy, we get another lower bound based on the total amount of processing:

$$C_{max}^{OPT}(L) \geq \frac{m\Delta + m(L_1 - b) + (b - \Delta) + p_n}{m}.$$

Adding up the two lower bounds above, we get

$$2C_{max}^{OPT}(L) \geq \frac{2mL_1 - (m-1)b + 2p_n}{m}$$

Because $r_n \geq b$, we also have

$$C_{max}^{OPT}(L) \geq b + p_n.$$

Hence we derive

$$\begin{aligned} \frac{C_{max}^{LS}(L)}{C_{max}^{OPT}(L)} &= \frac{L_1 + p_n}{C_{max}^{OPT}(L)} \\ &\leq \frac{2mC_{max}^{OPT}(L) + 2(m-1)(b + p_n)}{2mC_{max}^{OPT}(L)} \\ &\leq 2 - \frac{1}{m}. \end{aligned}$$

Hence we have $R(m, LS) \leq 2 - \frac{1}{m}$. This creates a contradiction as L is a counterexample satisfying $\frac{C_{max}^{LS}(L)}{C_{max}^{OPT}(L)} > 2 - \frac{1}{m}$. It is also well-known in Graham (1969) that, when $r_1 = r_2 = \ldots = r_n = 0$, the bound is tight. Hence (1) holds.

However, for jobs with arbitrary release times, (1) does not hold any more, which is stated in the next theorem.

Theorem 2. For the problem of scheduling jobs with arbitrary release times,

$$R(m, LS) = 3 - \frac{1}{m}.$$

Proof: Let $L = \{ J_1, J_2, \ldots, J_n \}$ be an arbitrary sequence of jobs. Job J_j has release time r_j and running time p_j ($j = 1, 2, \ldots, n$). Without loss of generality, we suppose that the scheduled completion time of job J_n is the largest job completion time for all the machines, i.e. the makespan. Let P be $\sum_{j=1}^{n-1} p_j$, u_i ($i = 1, 2, \ldots, m$) be the total idle time of machine M_i, and s be the starting time of job J_n. Let $u = s - L_1$, then we have

$$C_{max}^{LS}(L) = L_1 + p_n + u, \quad u \leq r_n, \quad u + p_n \leq r_n + p_n.$$

It is obvious that

$$r_n + p_n \leq C_{max}^{OPT}(L), \quad P + p_n \leq mC_{max}^{OPT}(L).$$

Because $C_{max}^{OPT}(L) \geq \max\{r_1, r_2, \ldots, r_n\}$ we have $u_i \leq C_{max}^{OPT}(L)$ ($i = 1, 2, \ldots, m$). So

$$\begin{aligned}
\frac{C_{max}^{LS}(L)}{C_{max}^{OPT}(L)} &= \frac{L_1 + p_n + u}{C_{max}^{OPT}(L)} \leq \frac{\sum_{i=1}^{m}(L_i + p_n) + mu}{mC_{max}^{OPT}(L)} \\
&= \frac{P + p_n + \sum_{i=1}^{m} u_i + (m-1)(u + p_n)}{mC_{max}^{OPT}(L)} \\
&\leq \frac{P + p_n + \sum_{i=1}^{m} u_i + (m-1)(r_n + p_n)}{mC_{max}^{OPT}(L)} \\
&\leq \frac{mC_{max}^{OPT}(L) + mC_{max}^{OPT}(L) + (m-1)C_{max}^{OPT}(L)}{mC_{max}^{OPT}(L)} \\
&= 3 - \frac{1}{m}.
\end{aligned}$$

By the arbitrariness of L we have $R(m, LS) \leq 3 - \frac{1}{m}$. The following example shows that the bound of $3 - \frac{1}{m}$ is tight.

Let $\overline{L}_0 = \{J_1, J_2, \cdots, J_{2m^2-m+1}\}$ with

$$\begin{aligned}
&r_j = i - \varepsilon, \quad p_j = \varepsilon, && (i-1)m + 1 \leq j \leq im, \quad i = 1, 2, \cdots, m; \\
&r_j = 0, \quad p_j = 1, && m^2 + 1 \leq j \leq 2m^2 - m; \\
&r_j = 0, \quad p_j = m, && j = 2m^2 - m + 1.
\end{aligned}$$

It is easy to see that the *LS* schedule is

$$\begin{aligned}
&M_i^{LS} = (J_i, J_{i+m}, \cdots, J_{i+(m-1)m}, J_{i+m^2}, \cdots, J_{i+(2m-2)m}), \quad i = 2, 3, \cdots, m, \\
&M_1^{LS} = (J_1, J_{1+m}, \cdots, J_{1+(m-1)m}, J_{1+m^2}, \cdots, J_{1+(2m-2)m}, J_{2m^2-m+1}).
\end{aligned}$$

Thus $C_{max}^{LS}(\overline{L}_0) = L_1 = m + m - 1 + m = 3m - 1$. One optimal off-line schedule is

$$\begin{aligned}
&M_i^{OPT} = (J_{m^2+(i-1)m+1}, J_{m^2+(i-1)m+2}, \cdots, J_{m^2+(i-1)m+m}, J_i, J_{i+m}, \cdots, J_{i+(m-1)m}), \\
&for \quad i = 1, 2, \cdots, m-1, \\
&M_m^{OPT} = (J_{2m^2-m+1}, J_m, J_{m+m}, \cdots, J_{m+(m-1)m}).
\end{aligned}$$

Thus $C_{max}^{OPT}(\overline{L}_0) = m + m\varepsilon$. Hence

$$R(m, LS) \geq \frac{C_{max}^{LS}(\overline{L}_0)}{C_{max}^{OPT}(\overline{L}_0)} = 3 - \frac{1 + 3m\varepsilon}{m + m\varepsilon}.$$

Let ε tend to zero, we have $R(m, LS) \geq 3 - \frac{1}{m}$. That means $R(m, LS) = 3 - \frac{1}{m}$.

The following theorem says that no on-line algorithm can have a worst case performance ratio better than 2 when jobs' release times are arbitrary.

Theorem 3. For scheduling jobs with arbitrary release times, there is no on-line algorithm with worst case ratio less then 2.

Proof. Suppose that algorithm A has worst case ratio less than 2. Let $L = \{J_1, J_2, \ldots, J_{m+1}\}$, with $r_1 = 1$, $p_1 = \varepsilon$, $r_j = 0$, $p_j = S + \varepsilon$ ($j = 2, 3, \ldots, m + 1$), where $S \geq 1$ is the starting processing time of

job J_1. Let $L^{(k)} = \{J_1, \ldots, J_k\}$ ($k = 1, 2, \ldots, m+1$). Because $R(m, A) < 2$, any two jobs from the job set $L^{(m)}$ cannot be assigned to the same machine and also $C^A_{max}(L^{(m+1)}) \geq 2(S+\varepsilon)$. But $C^{OPT}_{max}(L^{(m+1)}) = S + 2\varepsilon$, so

$$\frac{C^A_{max}(L^{(m+1)})}{C^{OPT}_{max}(L^{(m+1)})} \geq \frac{2(S+\varepsilon)}{S+2\varepsilon}.$$

Let ε tend to zero, we get $R(m, A) \geq 2$, which leads to a contradiction.
From the conclusions of Theorem 2 and Theorem 3, we know that algorithm *LS* is optimal for $m = 1$.

4. Improved Algorithms for $m \geq 2$

For $m \geq 2$, to bring down the performance ratio, Li & Huang (2004) introduced a modified *LS* algorithm, *MLS*, which satisfies $R(m, MLS) \leq 3 - \frac{1}{m} - \varepsilon_m$ with $\varepsilon_m \geq \varepsilon > 0$. To describe the algorithm, we let

$$\delta_m = \frac{3m-1}{m} - \frac{\lfloor \frac{m}{2} \rfloor}{m},$$
$$\eta_m = \frac{m}{\lfloor \frac{m}{2} \rfloor},$$

where $\lfloor x \rfloor$ denotes the largest integer not bigger than x.
In *MLS*, two real numbers ρ_m and τ_m will be used. They satisfy $2 + \tau_m < \rho_m < \frac{3m-1}{m}$ and $\tau_m > 0$, where $\tau_m = \frac{1}{\eta_m(\rho_m - \delta_m)} - 1$ and ρ_m is a root of the following equation

$$\frac{2[x - 1 + \eta_m(x - \delta_m)]}{2\eta_m^2(x-\delta_m)^2 + [\eta_m x(x-\delta_m) - 1][x - 1 + \eta_m(x-\delta_m)]} + \frac{2m-1}{m} = x.$$

Algorithm *MLS*

Step 1. Assume that L_i is the scheduled completion time of machine M_i ($i = 1, 2, \ldots, m$). Reorder machines such that $L_1 \leq L_2 \leq \ldots \leq L_m$ and let J_n be a new job given to the algorithm. Set $L_{m+1} = +\infty$.

Step 2. Suppose that 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 assign job J_n to machine M_i to be processed starting at time $\max\{T_1, r_n\}$ in the idle interval. If no such M_i exists, go to step 3.

Step 3. If $r_n \leq L_1$, we assign J_n on machine M_1 to start at time L_1.

Step 4. If $L_k < r_n \leq L_{k+1}$ for some $1 \leq k \leq m$ and $p_n \geq \tau_m r_n$, then we assign J_n on machine M_k to start at time r_n.

Step 5. If $L_k < r_n \leq L_{k+1}$ for some $1 \leq k \leq m$ and $p_n < \tau_m r_n$ and $L_{k+1} + p_n \leq \rho_m (r_n + p_n)$, then we assign J_n on machine M_{k+1} to start at time L_{k+1}.

Step 6. If $L_k < r_n \leq L_{k+1}$ for some $1 \leq k \leq m$ and $p_n < \tau_m r_n$ and $L_{k+1} + p_n > \rho_m (r_n + p_n)$, then we assign J_n on machine M_k to start at time r_n.

The following theorem was proved in Li & Huang (2004).

Theorem 4. For any $m \geq 2$, we have

$$R(m, MLS) \leq \rho_m.$$

Furthermore, there exists a fixed positive number ε independent of m such that

$$\rho_m \leq 3 - \frac{1}{m} - \varepsilon \leq 2.9392.$$

Another better algorithm, *NLMS*, was further proposed by Li & Huang (2007). In the following we will describe it in detail and reproduce the proofs from Li & Huang (2007). In the description, three real numbers α_m, β_m and ξ_m will be used, where $1 < \beta_m < 2$, $\xi_m\alpha_m < \frac{3m-1}{m}$ and they are the roots of the next three equations.

$$\begin{cases} \frac{2y^2}{2+(x-y)y} = 1 & (2) \\ zx = \frac{\lfloor \frac{m}{2} \rfloor}{my} + \frac{m-\lfloor \frac{m}{2} \rfloor}{m} + \frac{2m-1}{m} & (3) \\ \frac{2y^2}{2+y+(zx-2)y^2} = 1. & (4) \end{cases}$$

The next lemma establishes the existence of the three numbers and relate them to figures.

Lemma 5. There exist $x = \alpha_m$, $y = \beta_m$ and $z = \xi_m$ satisfying equations (2), (3) and (4) with $1 < \beta_m < 2$, $\beta_m < \alpha_m$, $\xi_m\alpha_m < \frac{3m-1}{m}$, $2 < \beta_m^2 < \beta_m + 2$ and $2 < \xi_m\alpha_m \leq 2.78436$ for any $m \geq 2$.

Proof. By equation (3) and (4), we have

$$\frac{\lfloor \frac{m}{2} \rfloor}{m} + \frac{1}{m} - \frac{\lfloor \frac{m}{2} \rfloor}{my} - \frac{2}{y^2} - \frac{1}{y} + 1 = 0. \quad (5)$$

Let $f(y) = \frac{\lfloor \frac{m}{2} \rfloor}{m} + \frac{1}{m} - \frac{\lfloor \frac{m}{2} \rfloor}{my} - \frac{2}{y^2} - \frac{1}{y} + 1$. It is easy to check that

$$f'(y) > 0, \quad f(1) < 0 \quad and \quad f(2) = \frac{\lfloor \frac{m}{2} \rfloor}{2m} + \frac{1}{m} > 0.$$

Hence there exists exactly one real number $1 < \beta_m < 2$ satisfying equation (5).
By equation (2), we have

$$\alpha_m = 3\beta_m - \frac{2}{\beta_m} > 3\beta_m - 2 > \beta_m,$$

where the two inequalities result from $\beta_m > 1$.
By equation (3), we get $\xi_m = \frac{1}{\alpha_m}(\frac{\lfloor \frac{m}{2} \rfloor}{m\beta_m} + \frac{m-\lfloor \frac{m}{2} \rfloor}{m} + \frac{2m-1}{m})$ and

$$\begin{aligned} \xi_m\alpha_m &= \frac{\lfloor \frac{m}{2} \rfloor}{m\beta_m} + \frac{m - \lfloor \frac{m}{2} \rfloor}{m} + \frac{2m-1}{m} \\ &< \frac{\lfloor \frac{m}{2} \rfloor}{m} + \frac{m - \lfloor \frac{m}{2} \rfloor}{m} + \frac{2m-1}{m} \\ &= 3 - \frac{1}{m}. \end{aligned}$$

Let $g(y,q) = \frac{q}{2q+1} + \frac{1}{2q+1} - \frac{q}{(2q+1)y} - \frac{2}{y^2} - \frac{1}{y} + 1$. It is easy to show that $\frac{\partial g(y,q)}{\partial q} < 0$ and $\frac{\partial g(y,q)}{\partial q} < 0$. Because $g(\beta_{2q+1}, q) = 0$, we have $\frac{d\beta_{2q+1}}{dq} > 0$. Because of equation (5), we get

$$3(\lim_{q\to+\infty} \beta_{2q+1})^2 - 3\lim_{q\to+\infty} \beta_{2q+1} - 4 = 0.$$

Hence $\lim_{q\to+\infty}\beta_{2q+1}$ = 1.75831. In addition, by equation (5), we have β_3 = 1.56619. Noticing that β_{2q+1} is an increasing function on q as $\frac{d\beta_{2q+1}}{dq} > 0$, we have

$$2 < \beta_3^2 \le \beta_{2q+1}^2;$$

$$\beta_{2q+1}^2 - \beta_{2q+1} - 2 < (\lim_{q\to+\infty}\beta_{2q+1})^2 - \lim_{q\to+\infty}\beta_{2q+1} - 2 < 0.$$

That means $2 < \beta_{2q+1}^2 < \beta_{2q+1} + 2$ holds. In the same way as above, we can show that $2 < \beta_{2q}^2 < \beta_{2q} + 2$ holds. Thus $2 < \beta_m^2 < \beta_m + 2$ holds for any $m \ge 2$.

By equation (4), we have $\xi_{2q+1}\alpha_{2q+1} = 4 - \frac{2}{\beta_{2q+1}^2} - \frac{1}{\beta_{2q+1}}$ and hence $\frac{d(\xi_{2q+1}\alpha_{2q+1})}{dq} = (\frac{4}{\beta_{2q+1}^3} + \frac{1}{\beta_{2q+1}^2})\frac{d\beta_{2q+1}}{dq} > 0$. Thus we get $\xi_{2q+1}\alpha_{2q+1} \ge \xi_3\alpha_3 > 2$ and

$$\begin{aligned}\xi_{2q+1}\alpha_{2q+1} &\le \lim_{q\to+\infty}\xi_{2q+1}\alpha_{2q+1}\\ &= 4 - \frac{2}{(\lim_{q\to+\infty}\beta_{2q+1})^2} - \frac{1}{\lim_{q\to+\infty}\beta_{2q+1}}\\ &= 2.78436,\end{aligned}$$

i.e. $2 < \xi_{2q+1}\alpha_{2q+1} \le 2.78436$. Similarly we can get $2 < \xi_{2q}\alpha_{2q} \le 2.78436$. That means $2 < \xi_m\alpha_m \le 2.78436$ holds for any $m \ge 2$.

For simplicity of presentation, in the following we drop the indices and write the three numbers as α, β and ξ if no confusion arises. The algorithm *NMLS* can be described as follows:

Algorithm *NMLS*

Step 0. $R_i^{(0)} := 0$, $L_i := 0$, $i = 1, 2, \dots, m$. $L_{m+1} := +\infty$.

Step 1. Assume that L_i is the scheduled completion time of machine M_i after job J_{n-1} is assigned. Reorder machines such that $L_1 \le L_2 \le \dots \le L_m$. $R^{(n-1)}(s)$ ($s = 1, 2, \dots, m$) represents the sth smallest number of $R_i^{(n-1)}$, $i = 1, 2, \dots, m$. Let J_n be a new job given to the algorithm.

Step 2. Suppose that 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 assign job J_n on machine M_i to start at time $\max\{T_1, r_n\}$ in the idle interval. $R_i^{(n)} := R_i^{(n-1)}$, $i = 1, 2, \dots, m$. If no such M_i exists, go to Step 3.

Step 3. If $r_n < L_1$, we assign J_n on machine M_1 to start at time L_1. $R_i^{(n)} := R_i^{(n-1)}$, $i = 1, 2, \dots, m$.

Step 4. If $L_k \le r_n < L_{k+1}$ and all of the following conditions hold:

(a) $R^{(n-1)}(\lfloor \frac{m}{2} \rfloor + 1) > \max\{L_k, \frac{R^{(n-1)}(m)}{\beta}\}$,

(b) $r_n > \frac{R^{(n-1)}(m)}{\beta}$,

(c) $p_n \le (\beta - 1)r_n$,

(d) $L_{k+1} + p_n \le \alpha(r_n + p_n)$,

then we assign J_n on machine M_{k+1} to start at time L_{k+1} and set $R_i^{(n)} := R_i^{(n-1)}$, $i = 1, 2, \dots, m$. Otherwise go to Step 5.

Step 5. Assign job J_n on machine M_k to start at time r_n. Set $R_k^{(n)} := r_n, R_i^{(n)} := R_i^{(n-1)}, i \ne k$.

5. Performance ratio analysis of algorithm NMLS

In the rest of this section, the following notation will be used: For any $1 \leq j < n$, $1 \leq i \leq m$, we use $L^{(j)}$ to denote the job list $\{J_1, J_2, \ldots, J_j\}$ and $L_i^{(j)}$ to denote the completion time of machine M_i before job J_j is assigned. For a given job list L, we set

$$U(L) = \sum_{i=1}^{m} u_i(L),$$

where $u_i(L)$ ($i = 1, 2, \ldots, m$) is the total idle time of machine M_i when job list L is scheduled by algorithm *NMLS*. We first observe the next two simple inequalities which will be used in the ratio analysis.

$$C_{max}^{OPT}(L) \geq r_j + p_j, \quad \text{for any} \quad j = 1, 2, \cdots, n, \tag{6}$$

$$C_{max}^{OPT}(L) > u_i(L), \quad i = 1, 2, \cdots, m. \tag{7}$$

Also if there exists a subset $\{J_{j_1}, \cdots, J_{j_q}\}$ in job list L satisfying $r_{j_s} \geq \overline{r}(s = 1, 2, \cdots, q)$, then the next inequality holds.

$$C_{max}^{OPT}(L) \geq \overline{r} + \frac{\sum_{s=1}^{q} p_{j_s}}{m}. \tag{8}$$

In addition, if $j_1 > j_2$, then

$$L_i^{(j_1)} \geq L_i^{(j_2)}, \quad R_i^{(j_1)} \geq R_i^{(j_2)}, \quad 1 \leq i \leq m.$$

In order to estimate *U(L)*, we need to consider how the idle time is created. For a new job J_n given to algorithm *NMLS*, if it is assigned in Step 5, then a new idle interval $[L_k, r_n]$ is created. If it is assigned in Step 3 or Step 4, no new idle time is created. If it is assigned in Step 2, new idle intervals may appear, but no new idle time appears. Hence only when a job is assigned in Step 5 can it make the total sum of idle time increase. Because of this fact, we will say idle time is created only by jobs which are assigned in Step 5. We further define the following terminologies

- A job J is referred to as an idle job on machine M_i, $1 \leq i \leq m$, if it is assigned on machine M_i in Step 5. An idle job J is referred to as a last idle job on machine M_i, $1 \leq i \leq m$, if J is assigned on machine M_i and there is no idle job on machine M_i after job J.

In the following, for any machine M_i, we will use $J_{\bar{i}_1}$ to represent the last idle job on machine M_i if there exist idle jobs on machine M_i, otherwise $J_{\bar{i}_1}$ to represent the first job (which starts at time 0) assigned on machine M_i.
Next we set

$$R = \max\{r_{\bar{i}_1} + p_{\bar{i}_1} | 1 \leq i \leq m\}; \qquad A = \{i | r_{\bar{i}_1} \geq \frac{R}{\beta}\}.$$

By the definitions of our notation, it is easy to see that the following facts are true:

$$R_i^{(n)} = r_{\bar{i}_1}, \quad i = 1, 2, \cdots, m;$$
$$R > R^{(n)}(m);$$
$$R \le C_{max}^{OPT}(L).$$

For the total idle time *U(L),* the next lemma provides an upper bound.

Lemma 6. For any job list $L = \{J_1, J_2, \ldots, J_n\}$, we have

$$\frac{U(L)}{mC_{max}^{OPT}(L)} \le \frac{\lfloor \frac{m}{2} \rfloor}{m\beta} + \frac{m - \lfloor \frac{m}{2} \rfloor}{m}.$$

Proof. By the definition of *R,* no machine has idle time later than time point *R*. We will prove this lemma according to two cases.

Case 1. At most $m - \lfloor \frac{m}{2} \rfloor$ machines in *A* are idle simultaneously in any interval [a, *b*] with $\frac{R}{\beta} \le$ a < *b*.

Let v_i be the sum of the idle time on machine M_i before time point $\frac{R}{\beta}$ and v_i' be the sum of the idle time on machine M_i after time point $\frac{R}{\beta}$, i = 1, 2, ... , m. The following facts are obvious:

$$u_i(L) = v_i + v_i', \qquad v_i \le \frac{R}{\beta}, \qquad 1 \le i \le m;$$
$$v_i' = 0, \qquad \forall i \notin A.$$

In addition, we have

$$\sum_{i \in A} v_i' \le (m - \lfloor \frac{m}{2} \rfloor)(1 - \frac{1}{\beta})R$$

because at most $m - \lfloor \frac{m}{2} \rfloor$ machines in *A* are idle simultaneously in any interval [a, *b*] with $\frac{R}{\beta} \le$ a < *b* $\le$ R. Thus we have

$$\begin{aligned}
\frac{U(L)}{mC_{max}^{OPT}(L)} &= \sum_{i=1}^{m} \frac{u_i(L)}{mC_{max}^{OPT}(L)} \\
&= \sum_{i=1}^{m} \frac{v_i}{mC_{max}^{OPT}(L)} + \sum_{i \in A} \frac{v_i'}{mC_{max}^{OPT}(L)} \\
&\le \sum_{i=1}^{m} \frac{R}{m\beta C_{max}^{OPT}(L)} + \frac{(m - \lfloor \frac{m}{2} \rfloor)(1 - \frac{1}{\beta})R}{mC_{max}^{OPT}(L)} \\
&\le \sum_{i=1}^{m} \frac{C_{max}^{OPT}(L)}{m\beta C_{max}^{OPT}(L)} + \frac{(m - \lfloor \frac{m}{2} \rfloor)(1 - \frac{1}{\beta})C_{max}^{OPT}(L)}{mC_{max}^{OPT}(L)} \\
&= \frac{\lfloor \frac{m}{2} \rfloor}{m\beta} + \frac{m - \lfloor \frac{m}{2} \rfloor}{m}.
\end{aligned}$$

Case 2. At least $m - \lfloor \frac{m}{2} \rfloor + 1$ machines in A are idle simultaneously in an interval $[a, b]$ with $\frac{R}{\beta} \leq a < b$.
In this case, we select a and b such that at most $m - \lfloor \frac{m}{2} \rfloor$ machines in A are idle simultaneously in any interval $[a', b']$ with $a < b \leq a' < b'$. Let

$$\overline{A} = \{i \in A | M_i \text{ is idle in } [a, b]\}.$$

That means $|\overline{A}| > m - \lfloor \frac{m}{2} \rfloor$ by our assumption. Let M_{i_0}, $i_0 \in \overline{A}$, be such a machine that its idle interval $[a, b]$ is created last among all machines $M_i, i \in \overline{A}$. Let

$$A' = \overline{A} \setminus \{i_0\}.$$

Suppose the idle interval $[a, b]$ on machine M_{i_0} is created by job $J_{\bar{i}_0}$. That means that the idle interval $[a, b]$ on machine M_i for any $i \in A'$ has been created before job $J_{\bar{i}_0}$ is assigned. Hence we have $R_i^{(\bar{i}_0 - 1)} \geq b > a \geq L_{i_0}^{(\bar{i}_0)}$ for any $i \in A'$. In the following, let

$$r_{\bar{k}_1} = \min\{r_{\bar{i}_0}, \min\{r_{\bar{i}_1} | i \in A'\}\}.$$

We have $r_{\bar{k}_1} \geq b$ because $r_{\bar{i}_0} \geq b$ and $r_{\bar{i}_1} \geq b$, $\forall i \in A'$.
What we do in estimating $C_{max}^{OPT}(L)$ is to find a job index set S such that each job J_j ($j \in S$) satisfies $r_j \geq \frac{r_{\bar{k}_1}}{\beta}$ and $\sum_{j \in S} p_j \geq |A'|(\alpha - \beta) r_{\bar{k}_1}$. And hence by (8) we have

$$C_{max}^{OPT}(L) \geq \frac{r_{\bar{k}_1}}{\beta} + \frac{|A'|(\alpha - \beta) r_{\bar{k}_1}}{m}.$$

To do so, we first show that

$$L_i^{(\bar{i}_0)} + p_{\bar{i}_0} > \alpha(r_{\bar{i}_0} + p_{\bar{i}_0}), \quad i \in A' \tag{9}$$

holds. Note that job $J_{\bar{i}_0}$ must be assigned in Step 5 because it is an idle job. We can conclude that (9) holds if we can prove that job $J_{\bar{i}_0}$ is assigned in Step 5 because the condition (d) of Step 4 is violated. That means we can establish (9) by proving that the following three inequalities hold by the rules of algorithm *NMLS:*

(a) $R^{(\bar{i}_0 - 1)}(\lfloor \frac{m}{2} \rfloor + 1) > \max\{L_{i_0}^{(\bar{i}_0)}, \frac{R^{(\bar{i}_0 - 1)}(m)}{\beta}\}$;

(b) $r_{\bar{i}_0} > \frac{R^{(\bar{i}_0 - 1)}(m)}{\beta}$;

(c) $p_{\bar{i}_0} \leq (\beta - 1) r_{\bar{i}_0}$.

The reasoning for the three inequalities is:
(a). As $m - |A'| + 1 = m - |\overline{A}| + 2 \leq m - (m - \lfloor \frac{m}{2} \rfloor) + 1 = \lfloor \frac{m}{2} \rfloor + 1$ we have

$$\begin{aligned} R^{(\bar{i}_0 - 1)}(\lfloor \frac{m}{2} \rfloor + 1) &\geq R^{(\bar{i}_0 - 1)}(m - |A'| + 1) \geq \min_{i \in A'}\{R_i^{(\bar{i}_0 - 1)}\} \\ &\geq b > a \geq \frac{R^n(m)}{\beta} \geq \frac{R^{(\bar{i}_0 - 1)}(m)}{\beta}. \end{aligned}$$

Next we have $a \geq L_{i_0}^{(\bar{i}_0)}$ because idle interval [a, b] on machine M_{i_0} is created by job $J_{\bar{i}_0}$. Hence we have

$$R^{(\bar{i}_0-1)}(\lfloor \frac{m}{2} \rfloor + 1) > a \geq \max\{L_{i_0}^{(\bar{i}_0)}, \frac{R^{(\bar{i}_0-1)}(m)}{\beta}\},$$

i.e. the first inequality is proved.

(b). This follows because $r_{\bar{i}_0} \geq b > a \geq \frac{R^{(\bar{i}_0-1)}(m)}{\beta}$.

(c). As $\frac{r_{\bar{i}_0}+p_{\bar{i}_0}}{\beta} \leq \frac{R}{\beta} \leq a < b \leq r_{\bar{i}_0}$ we have $p_{\bar{i}_0} \leq (\beta-1)r_{\bar{i}_0}$.

For any $i \in A'$, by (9) and noticing that $r_{\bar{i}_1} < R \leq \beta a < \beta b \leq \beta r_{\bar{k}_1}$ and $\alpha > \beta > 1$, we have

$$L_i^{(\bar{i}_0)} - r_{\bar{i}_1} > \alpha(r_{\bar{i}_0} + p_{\bar{i}_0}) - p_{\bar{i}_0} - r_{\bar{i}_1} > \alpha r_{\bar{i}_0} - r_{\bar{i}_1} \geq \alpha r_{\bar{i}_0} - \beta r_{\bar{k}_1} \geq \alpha r_{\bar{k}_1} - \beta r_{\bar{k}_1} > 0.$$

That means job $J_{\bar{i}_1}$ appears before $J_{\bar{i}_0}$, i.e. $\bar{i}_0 > \bar{i}_1$. We set

$$S_i = \{j | J_j \text{ is processed in interval } [r_{\bar{i}_1}, L_i^{(\bar{i}_0)}] \text{ on machine } M_i\}, \forall i \in A';$$

$$S = \cup_{i \in A'} S_i.$$

We have $\sum_{j \in S_i} p_j = L_i^{(\bar{i}_0)} - r_{\bar{i}_1}$ because $J_{\bar{i}_1}$ is the last idle job on machine M_i for any $i \in A'$. Hence we have

$$\sum_{j \in S} p_j = \sum_{i \in A'} \sum_{j \in S_i} p_j = \sum_{i \in A'} (L_i^{(\bar{i}_0)} - r_{\bar{i}_1}) > \sum_{i \in A'} (\alpha r_{\bar{k}_1} - \beta r_{\bar{k}_1}) = |A'|(\alpha - \beta) r_{\bar{k}_1}. \quad (10)$$

Now we will show the following (11) holds:

$$r_j \geq \frac{r_{\bar{k}_1}}{\beta}, \quad \forall j \in S. \quad (11)$$

It is easy to check that $\bar{i}_1 \in S_i$ and $r_{\bar{i}_1} > \frac{r_{\bar{i}_1}}{\beta} \geq \frac{r_{\bar{k}_1}}{\beta}$ for any $i \in A'$, i.e. (11) holds for any $j \in S_i$ ($i \in A'$) and $j = \bar{i}_1$. For any $j \in S_i$ ($i \in A'$) and $j \neq \bar{i}_1$, we want to establish (11) by showing that J_j is assigned in Step 4. It is clear that job J_j is not assigned in Step 5 because it is not an idle job. Also $\bar{i}_0 > j$ because $L_i^{(j)} < L_i^{(\bar{i}_0)}$. Thus we have

$$L_i^{(j)} > r_{\bar{i}_1} \geq b > L_{i_0}^{(\bar{i}_0)} \geq L_{i_0}^{(j)},$$

where the first inequality results from $j \neq \bar{i}_1$ and the last inequality results from $\bar{i}_0 > j$. That means J_j is not assigned in Step 3 because job J_j is not assigned on the machine with the smallest completion time. In addition, observing that job $J_{\bar{i}_1}$ is the last idle job on machine M_i and $L_i^{(j)} > r_{\bar{i}_1}$ by the definition of S_i, we can conclude that J_j is assigned on machine M_i to start at time $L_i^{(j)}$. That means $j > \bar{i}_1$ and J_j cannot be assigned in Step 2. Hence J_j must be assigned in Step 4. Thus by the condition (b) in Step 4, we have

$$r_j > \frac{R^{(j-1)}(m)}{\beta} \geq \frac{r_{\bar{i}_1}}{\beta} \geq \frac{r_{\bar{k}_1}}{\beta},$$

where the second inequality results from $j > \bar{i}_1$. Summing up the conclusions above, for any $j \in S$, (11) holds. By (8), (10) and (11) we have

$$C_{max}^{OPT}(L) \geq \frac{r_{\bar{k}_1}}{\beta} + \frac{\sum_{j\in S} p_j}{m} \geq \frac{r_{\bar{k}_1}}{\beta} + \frac{|A'|(\alpha-\beta)r_{\bar{k}_1}}{m} = \frac{m+|A'|(\alpha-\beta)\beta}{m\beta} r_{\bar{k}_1}.$$

Now we begin to estimate the total idle time $U(L)$. Let $\bar{v}_i$ be the sum of the idle time on machine M_i before time point $r_{\bar{k}_1}$ and $\bar{v}'_i$ be the sum of the idle time on machine M_i after time point $r_{\bar{k}_1}$, $i = 1, 2, \ldots, m$. The following facts are obvious by our definitions:

$$u_i(L) = \bar{v}_i + \bar{v}'_i, \qquad \bar{v}_i \leq r_{\bar{k}_1}, \qquad 1 \leq i \leq m;$$
$$\bar{v}'_i = 0, \qquad \forall i \notin A.$$

By our definition of b and k_1, we have that $b \leq r_{\bar{k}_1}$ and hence at most $m - \lfloor \frac{m}{2} \rfloor$ machines in A are idle simultaneously in any interval $[a', b']$ with $r_{\bar{k}_1} \leq a' < b' \leq R$. Noting that no machine has idle time later than R, we have

$$\sum_{i\in A} \bar{v}'_i \leq (m - \lfloor \frac{m}{2} \rfloor)(R - r_{\bar{k}_1}) \leq (m - \lfloor \frac{m}{2} \rfloor)(\beta r_{\bar{k}_1} - r_{\bar{k}_1}) \leq |A'|(\beta r_{\bar{k}_1} - r_{\bar{k}_1}).$$

Thus we have

$$\begin{aligned}
\frac{U(L)}{mC_{max}^{OPT}(L)} &= \sum_{i=1}^{m} \frac{u_i(L)}{mC_{max}^{OPT}(L)} = \sum_{i=1}^{m} \frac{\bar{v}_i}{mC_{max}^{OPT}(L)} + \sum_{i\in A} \frac{\bar{v}'_i}{mC_{max}^{OPT}(L)} \\
&\leq \frac{mr_{\bar{k}_1}}{mC_{max}^{OPT}(L)} + \frac{|A'|(\beta r_{\bar{k}_1} - r_{\bar{k}_1})}{mC_{max}^{OPT}(L)} \leq \frac{|A'|\beta^2 + (m-|A'|)\beta}{m+|A'|(\alpha-\beta)\beta} \\
&\leq \frac{\beta(\beta+1)}{2+(\alpha-\beta)\beta} = \frac{2\beta^2}{2+(\alpha-\beta)\beta}[\frac{1}{2\beta} + \frac{1}{2}] \\
&= \frac{1}{2\beta} + \frac{1}{2} \leq \frac{\lfloor \frac{m}{2} \rfloor}{m\beta} + \frac{m - \lfloor \frac{m}{2} \rfloor}{m}.
\end{aligned}$$

The last inequality follows by observing that the function $h(x) = \frac{x}{m\beta} + \frac{m-x}{m}$ is a decreasing function of x for $x \in [\lfloor \frac{m}{2} \rfloor, \frac{m}{2}]$. The second inequality follows because $|A'| = |\overline{A}| - 1 \geq m - \lfloor \frac{m}{2} \rfloor \geq \frac{m}{2}$ and $g(x) = \frac{x\beta^2 + (m-x)\beta}{m + x(\alpha-\beta)\beta}$ is a decreasing function of x on $[\frac{m}{2}, |A'|]$. The fact that $g(x)$ is a decreasing function follows because $g'(x) < 0$ as

$$\alpha\beta - \beta^2 - \beta + 1 = (3\beta - \frac{2}{\beta})\beta - \beta^2 - \beta + 1 = (\beta - 1)(2\beta + 1) > 0.$$

The next three lemmas prove that $\xi\alpha$ is an upper bound for $\frac{C_{max}^{NMLS}(L)}{C_{max}^{OPT}(L)}$. Without loss of generality from now on, we suppose that the completion time of job J_n is the largest job completion time for all machines, i.e. the makespan $C_{max}^{NMLS}(L)$. Hence according to this assumption, J_n cannot be assigned in Step 2.

Lemma 7. If J_n is placed on M_k with $L_k \leq r_n < L_{k+1}$, then

$$\frac{C_{max}^{NMLS}(L)}{C_{max}^{OPT}(L)} \leq \xi\alpha.$$

Proof. This results from $C_{max}^{NMLS}(L) = r_n + p_n$ and $C_{max}^{OPT}(L) \geq r_n + p_n$.

Lemma 8. If J_n is placed on M_{k+1} with $L_k \leq r_n < L_{k+1}$, then

$$\frac{C_{max}^{NMLS}(L)}{C_{max}^{OPT}(L)} \leq \xi\alpha.$$

Proof. Because $C_{max}^{NMLS}(L) = L_{k+1} + p_n$ and $C_{max}^{OPT}(L) \geq r_n + p_n$, this lemma holds if $L_{k+1} + p_n \leq \xi\alpha(p_n + r_n)$.

Suppose $L_{k+1} + p_n > \xi\alpha(p_n + r_n)$. For any $1 \leq i \leq m$, let

$$\overline{S}_i = \{j | J_j \text{ is processed in interval } [R_i^{(n)}, L_i] \text{ on machine } M_i\}.$$

It is easy to see that

$$R_i^{(n-1)} = R_i^{(n)}, \quad i = 1, 2, \cdots, m \quad \text{and}$$
$$\sum_{j \in \overline{S}_i} p_j = L_i - R_i^{(n)} \geq L_{k+1} - R_i^{(n)}, \quad i = k+1, k+2, \cdots, m;$$

hold. Let

$$B = \{i | R_i^{(n)} \geq \frac{R^{(n)}(m)}{\beta}\};$$
$$r_{11} = \min\{r_n, \min\{R_i^{(n)} | i \in B\}\}.$$

By the rules of our algorithm, we have

$$R^{(n)}(\lfloor \frac{m}{2} \rfloor + 1) = R^{(n-1)}(\lfloor \frac{m}{2} \rfloor + 1) > \frac{R^{(n-1)}(m)}{\beta} = \frac{R^{(n)}(m)}{\beta};$$
$$r_n \geq \frac{R^{(n-1)}(m)}{\beta} = \frac{R^{(n)}(m)}{\beta}$$

because J_n is assigned in Step 4. Hence we have $|B| \geq m - \lfloor \frac{m}{2} \rfloor \geq \frac{m}{2}$ and $r_{11} \geq \frac{R^{(n)}(m)}{\beta}$. By the same way used in the proof of Lemma 6, we can conclude that the following inequalities hold for any $i \in B$:

$$r_j \geq \frac{R_i^{(n)}}{\beta} \geq \frac{r_{11}}{\beta}, \quad \forall j \in \overline{S}_i.$$

Thus by (8) and (10) we have

$$\begin{aligned} C_{max}^{OPT}(L) &\geq \frac{r_{11}}{\beta} + \sum_{i\in B} \frac{\sum_{j\in \overline{S}_i} p_j}{m} \\ &\geq \frac{r_{11}}{\beta} + \sum_{i\in B} \frac{L_{k+1} - R^{(n)}(m)}{m} \\ &\geq \frac{r_{11}}{\beta} + \frac{|B|(L_{k+1} - \beta r_{11})}{m} \\ &\geq \frac{2r_{11} + (L_{k+1} - \beta r_{11})\beta}{2\beta} \\ &\geq \frac{2r_n + (L_{k+1} - \beta r_n)\beta}{2\beta}. \end{aligned}$$

The second last inequality results from that $|B| \geq \frac{m}{2}$ and

$$L_{k+1} - \beta r_{11} > \xi\alpha(r_n + p_n) - p_n - \beta r_{11} > \xi\alpha r_{11} - \beta r_{11} > 0$$

as $\beta < 2 < \xi\alpha$. The last equality follows because $\beta^2 > 2$ and $r_n \geq r_{11}$. Also we have $\frac{r_n}{r_n+p_n} \geq \frac{1}{\beta}$ because J_n is assigned in Step 4. Hence we have

$$\begin{aligned} \frac{C_{max}^{NMLS}(L)}{C_{max}^{OPT}(L)} &= \frac{L_{k+1} + p_n}{C_{max}^{OPT}(L)} \\ &\leq \frac{2\beta(L_{k+1} + p_n)}{2r_n + (L_{k+1} + p_n - p_n - \beta r_n)\beta} \\ &\leq \frac{2\beta\xi\alpha(r_n + p_n)}{2r_n + (\xi\alpha(r_n + p_n) - p_n - \beta r_n)\beta} \\ &= \frac{2\beta\xi\alpha}{\frac{(2+\beta-\beta^2)r_n}{r_n+p_n} + (\xi\alpha - 1)\beta} \\ &\leq \frac{2\beta\xi\alpha}{\frac{2+\beta-\beta^2}{\beta} + (\xi\alpha - 1)\beta} \\ &= \frac{2\beta^2\xi\alpha}{2 + \beta + (\xi\alpha - 2)\beta^2} \\ &= \xi\alpha. \end{aligned}$$

The second inequality results from the fact that $f(x) = \frac{2\beta x}{2r_n + (x - p_n - \beta r_n)\beta}$ is a decreasing function of x for $x \geq 2r_n - (p_n + \beta r_n)\beta$ as $2 - \beta^2 < 0$. The last inequality results from $2 + \beta - \beta^2 > 0$ and the last equation results from equation (4).

Lemma 9. If job J_n is placed on machine M_1, then we have

$$\frac{C_{max}^{NMLS}(L)}{C_{max}^{OPT}(L)} \leq \xi\alpha.$$

Proof. In this case we have $L_1 \geq r_n$ and $C_{max}^{NMLS}(L) = L_1 + p_n$. Thus we have

$$\begin{aligned}
\frac{C_{max}^{NMLS}(L)}{C_{max}^{OPT}(L)} &= \frac{L_1 + p_n}{C_{max}^{OPT}(L)} \leq \frac{\sum_{i=1}^{m}(L_i + p_n)}{mC_{max}^{OPT}(L)} \\
&= \frac{\sum_{j=1}^{n} p_j + \sum_{i=1}^{m} u_i(L) + (m-1)p_n}{mC_{max}^{OPT}(L)} \\
&\leq \frac{\sum_{j=1}^{n} p_j + \sum_{i=1}^{m} u_i(L) + (m-1)(r_n + p_n)}{mC_{max}^{OPT}(L)} \\
&\leq \frac{mC_{max}^{OPT}(L) + \sum_{i=1}^{m} u_i(L) + (m-1)C_{max}^{OPT}(L)}{mC_{max}^{OPT}(L)} \\
&= \frac{U(L)}{mC_{max}^{OPT}(L)} + \frac{2m-1}{m} \\
&\leq \frac{\lfloor \frac{m}{2} \rfloor}{m\beta} + \frac{m - \lfloor \frac{m}{2} \rfloor}{m} + \frac{2m-1}{m} \\
&= \xi\alpha.
\end{aligned}$$

The next theorem proves that NMLS has a better performance than MLS for $m \geq 2$.

Theorem 10. For any job list L and $m \geq 2$, we have

$$R(m, NMLS) \leq \xi_m \alpha_m \leq 2.78436 \quad \text{and} \quad \xi_m \alpha_m < 3 - \frac{1}{m}.$$

Proof. By Lemma 5 and Lemma 7—Lemma 9, Theorem 10 is proved.

The comparison for some m among the upper bounds of the three algorithms' performance ratios is made in Table 1, where $R(m, LS) = 3 - \frac{1}{m}$.

m	α_m	β_m	$R(m, LS)$	$R(m, MLS)$	$R(m, NMLS)$
2	2.943	1.443	2.50000	2.47066	2.3465
3	3.42159	1.56619	2.66667	2.63752	2.54616
9	3.88491	1.68955	2.88889	2.83957	2.7075
12	3.89888	1.69333	2.91668	2.86109	2.71194
oo	4.13746	1.75831	3.00000	2.93920	2.78436

Table 1. A comparison of LS, MLS, and NMLS

6. LS scheduling for jobs with similar lengths

In this section, we extend the problem to be semi-online and assume that the processing times of all the jobs are within [l,*r*], where $r \geq 1$. We will analyze the performance of the *LS* algorithm. First again let L be the job list with n jobs. In the LS schedule, let L_i be the completion time of machine M_i and $u_{i1}, \ldots, u_{ik_i}$ denote all the idle time intervals of machine M_i ($i = 1, 2, \ldots, m$) just before J_n is assigned. The job which is assigned to start right after u_{ij} is denoted by J_{ij} with release time r_{ij} and processing time p_{ij}. By the definitions of u_{ij} and r_{ij}, it is easy to see that r_{ij} is the end point of u_{ij}. To simplify the presentation, we abuse the notation and use u_{ij} to denote the length of the particular interval as well.

The following simple inequalities will be referred later on.

$$p_n \le C_{max}^{OPT}(L), \quad mC_{max}^{OPT}(L) \ge \sum_{i=1}^{n} p_i + U, \tag{12}$$

$$C_{max}^{OPT}(L) \ge \sum_{j=1}^{k_i}(u_{ij} + p_{ij}), \quad i = 1, 2, \cdots, m, \tag{13}$$

$$\begin{aligned} \frac{L_1 + p_n}{C_{max}^{OPT}(L)} &\le \frac{\sum_{i=1}^{m}(L_i + p_n)}{mC_{max}^{OPT}(L)} \\ &= \frac{\sum_{j=1}^{n} p_j + \sum_{j=1}^{k_1} u_{1j} + \cdots + \sum_{j=1}^{k_m} u_{mj} + (m-1)p_n}{mC_{max}^{OPT}(L)}, \end{aligned} \tag{14}$$

where U is the total idle time in the optimal schedule.

The next theorem establishes an upper bound for LS when $m \ge 2$ and a tight bound when $m = 1$.

Theorem 11. For any $m \ge 2$, we have

$$R(m, LS) \le \begin{cases} 3 - \frac{1}{m} - \frac{1}{r} & r \ge \frac{m}{m-1} \\ 1 + \frac{2r}{1+2r} + \frac{(m-1)r}{m(1+2r)} & 1 \le r < \frac{m}{m-1} \end{cases} \tag{15}$$

and $R(1, LS) = 1 + \frac{r}{1+r}$.

We will prove this theorem by examining a minimal counter-example of (15). A job list L = { J_1, J_2, ... J_n} is called a minimal counter-example of (15) if (15) does not hold for L, but (15) holds for any job list L' with $|L'| < |L|$. In the following discussion, let L be a minimal counter-example of (15). It is obvious that, for a minimal counter-example L, the makespan is the completion time of the last job J_n, i.e. $L_1 + p_n$. Hence we have

$$\frac{C_{max}^{LS}(L)}{C_{max}^{OPT}(L)} = \frac{L_1 + p_n}{C_{max}^{OPT}(L)}.$$

We first establish the following Observation and Lemma 12 for such a minimal counter-example.

Observation. In the LS schedule, if one of the machines has an idle interval [0, T] with $T > r$, then we can assume that at least one of the machines is scheduled to start processing at time zero.

Proof. If there exists no machine to start processing at time zero, let δ be the earliest starting time of all the machines and $t_0 = \min\{\delta, T - r\}$. It is not difficult to see that any job's release time is at least t_0 because, if there exists a job with release time less than t_0, it would be assigned to the machine with idle interval [0, T] to start at its release time by the rules of LS. Now let L' be the job list which comes from list L by pushing forward the release time of each job to be t_0 earlier. Then L' has the same schedule as L for the algorithm LS. But the makespan of L' is t_0 less than the makespan of L not only for the LS schedule but also for the optimal schedule. Hence we can use L' as a minimal counter example and the observation holds for L'.

Lemma 12. There exists no idle time with length greater than $2r$ when $m \geq 2$ and there is no idle time with length greater than r when $m = 1$ in the LS schedule.

Proof. For $m \geq 2$ if the conclusion is not true, let $[T_1, T_2]$ be such an interval with $T_2 - T_1 > 2r$. Let L^0 be the job set which consists of all the jobs that are scheduled to start at or before time T_1. By Observation , L^0 is not empty. Let $\overline{L} = L \setminus L^0$. Then $\overline{L}$ is a counter-example too because $\overline{L}$ has the same makespan as L for the algorithm LS and the optimal makespan of $\overline{L}$ is not larger than that of L. This is a contradiction to the minimality of L. For $m = 1$, we can get the conclusion by employing the same argument.

Now we are ready to prove Theorem 11.

Proof. Let αr be the largest length of all the idle intervals. If $\alpha \leq \frac{r-1}{r}$, then by (12), (13) and (14) we have

$$\begin{aligned}
\frac{C_{max}^{LS}(L)}{C_{max}^{OPT}(L)} &\leq 1 + \frac{\sum_{i=1}^{k_1} u_{1i}}{m\sum_{i=1}^{k_1}(u_{1i}+p_{1i})} + \cdots + \frac{\sum_{i=1}^{k_m} u_{m_i}}{m\sum_{i=1}^{k_m}(u_{m_i}+p_{m_i})} + \frac{(m-1)p_n}{mC_{max}^{OPT}(L)} \\
&\leq 1 + \frac{\sum_{i=1}^{k_1} u_{1i}}{m\sum_{i=1}^{k_1}(u_{1i}+1)} + \cdots + \frac{\sum_{i=1}^{k_m} u_{m_i}}{m\sum_{i=1}^{k_m}(u_{m_i}+1)} + \frac{(m-1)}{m} \\
&\leq 1 + \frac{k_1\alpha r}{m(k_1+k_1\alpha r)} + \cdots + \frac{k_m\alpha r}{m(k_m+k_m\alpha r)} + \frac{(m-1)}{m} \\
&= 2 - \frac{1}{m} + \frac{\alpha r}{1+\alpha r} \leq 3 - \frac{1}{m} - \frac{1}{r}.
\end{aligned}$$

Next by use of $C_{max}^{OPT}(L) \geq 1 + \alpha r$ instead of $C_{max}^{OPT}(L) \geq p_n$ and observe that $p_n \leq r$ we have

$$\begin{aligned}
\frac{C_{max}^{LS}(L)}{C_{max}^{OPT}(L)} &\leq 1 + \frac{k_1\alpha r}{m(k_1+k_1\alpha r)} + \cdots + \frac{k_m\alpha r}{m(k_m+k_m\alpha r)} + \frac{(m-1)r}{m(1+\alpha r)} \\
&= 1 + \frac{\alpha r}{1+\alpha r} + \frac{(m-1)r}{m(1+\alpha r)}.
\end{aligned}$$

So if $m \geq 2$, r $\geq \frac{m}{m-1}$ and $\alpha \geq \frac{r-1}{r}$, we have

$$\frac{C_{max}^{LS}(L)}{C_{max}^{OPT}(L)} \leq 1 + \frac{\alpha r}{1+\alpha r} + \frac{(m-1)r}{m(1+\alpha r)}\Big|_{\alpha=\frac{r-1}{r}} = 3 - \frac{1}{m} - \frac{1}{r}$$

because $1 + \frac{\alpha r}{1+\alpha r} + \frac{(m-1)r}{m(1+\alpha r)}$ is a decreasing function of α. Hence the conclusion for $m \geq 2$ and r $\geq \frac{m}{m-1}$ is proved. If $m \geq 2$ and $r < \frac{m}{m-1}$ we have

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

because $\alpha \leq 2$ by Lemma 12 and $1 + \frac{\alpha r}{1+\alpha r} + \frac{(m-1)r}{m(1+\alpha r)}$ is an increasing function of α. Hence the conclusion for $m \geq 2$ is proved. For $m = 1$ we have

$$\frac{C_{max}^{LS}(L)}{C_{max}^{OPT}(L)} \leq 1 + \frac{\alpha r}{1+\alpha r} + \frac{(m-1)r}{m(1+\alpha r)} \leq 1 + \frac{r}{1+r}$$

because α < 1 by Lemma 12. Consider $L = \{J_1, J_2\}$ with $r_1 = r - \varepsilon$,$p_1 = 1$, $r_2 = 0$, $p_2 = r$ and let ε tend to zero. Then we can show that this bound is tight for $m = 1$.
From Theorem 11, for $m \geq 2$ and $1 \leq r < \frac{m}{m-1}$ we have $R(m, LS) < 2$ because $1+\frac{2r}{1+2r}+\frac{(m-1)r}{m(1+2r)}$ is an increasing function of r and $1 + \frac{2r}{1+2r} + \frac{(m-1)r}{m(1+2r)}|_{r=\frac{m}{m-1}} = 2$. This is significant because no online algorithm can have a performance ratio less than 2 as stated in Theorem 3. An interesting question for the future research is then how to design a better algorithm than LS for this semi-online scheduling problem. The next theorem provides a lower bound of any on-line algorithm for jobs with similar lengths when $m = 1$.
Theorem 13. For $m = 1$ and any algorithm A for jobs with lengths in $[1, r]$, we have

$$R(1, A) \geq 1 + \frac{\alpha}{1+r},$$

where α satisfies the following conditions:
a) $\frac{1+r+\alpha}{1+r} = \frac{1+\beta}{1+\alpha} = \frac{1+r+\beta}{1+\beta}$;
b) $\alpha < r < \beta$.
Proof. Let job J_1 be the first job in the job list with $p_1 = 1$ and $r_1 = \alpha$. Assume that if J_1 is assigned by algorithm A to start at any time in $[\alpha, r)$, then the second job J_2 comes with $p_2 = r$ and $r_2 = 0$. Thus for these two jobs, $C_{max}^{A} \geq 1 + r + \alpha$ and $C_{max}^{OPT} = 1 + r$. Hence we get

$$R(1, A) \geq \frac{1+r+\alpha}{1+r}.$$

On the other hand, if J_1 is assigned by algorithm A to start at any time k, $k \in [r, \beta)$, then the second job J_2 comes with $p_2 = r$ and $r_2 = k - r + \varepsilon$. Thus for these two jobs, $C_{max}^{A} \geq 1 + r + k$ and $C_{max}^{OPT} = 1 + k + \varepsilon$. Hence we get

$$R(1, A) \geq \frac{1+r+k}{1+k+\varepsilon}.$$

Let ε tend to zero, we have

$$R(1, A) \geq \frac{1+r+k}{1+k} \geq \frac{1+r+\beta}{1+\beta},$$

where the second inequality results from the fact that $\frac{1+r+x}{1+x}$ is a decreasing function of x for $x \geq 0$. Lastly assume that if J_1 is assigned by algorithm A to start at any time after β, then no other job comes. Thus for this case, $C_{max}^{A} \geq 1 + \beta$ and $C_{max}^{OPT} = 1 + \alpha$. Hence we get

$$R(1, A) \geq \frac{1+\beta}{1+\alpha}.$$

For $r = 1$, we get $\alpha = 0.7963$ and hence $R(\text{l}, \text{A}) \geq 1.39815$. Recall from Theorem 11, $R(\text{l}, LS) = 1.5$ when $r = 1$. Therefore *LS* provides a schedule which is very close to the lower bound.

7. References

Albers, S. (1999) Better bounds for online scheduling. *SIAM J. on Computing*, Vol.29, 459-473.

Bartal, Y., Fiat, A., Karloff, H. & Vohra, R. (1995) New algorithms for an ancient scheduling problem. *J. Comput. Syst. Sci*., Vol.51(3), 359-366.

Chen, B., Van Vliet A., & Woeginger, G. J. (1994) New lower and upper bounds for on-line scheduling. *Operations Research Letters,* Vol.16, 221-230.

Chen, B. & Vestjens, A. P. A. (1997) Scheduling on identical machines: How good is LPT in an on-line setting? *Operations Research Letters*, Vol.21, 165-169.

Dosa, G., Veszprem, & He, Y. (2004) Semi-online algorithms for parallel machine scheduling problems. *Computing*, Vol.72, 355-363.

Faigle, U., Kern, W., & Turan, G. (1989) On the performance of on-line algorithms for partition problems. *Act Cybernetica*, Vol.9, 107-119.

Fleischer R. & Wahl M. (2000) On-line scheduling revisited. *Journal of Scheduling*. Vol.3, 343-353.

Galambos, G. & Woeginger, G. J. (1993) An on-line scheduling heuristic with better worst case ratio than Graham's List Scheduling. *SIAM J. Comput*. Vol.22, 349-355.

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

Hall, L. A. & Shmoys, D. B. (1989) Approximation schemes for constrained scheduling problems. Proceedings of 30th Ann. *IEEE Symp. on foundations of computer science*, IEEE Computer Society Press, Loss Alamitos, CA, 134-139.

He, Y., Jiang, Y., & Zhou, H. (2007) Optimal Preemptive Online Algorithms for Scheduling with Known Largest Size on two Uniform Machines, *Acta Mathematica Sinica*, Vol.23, 165-174.

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

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

Karger, D. R., Philips, S. J., & Torng, E. (1996) A better algorithm for an ancient scheduling problem. *J. of Algorithm*, vol.20, 400-430.

Kellerer, H. (1991) Bounds for non-preemptive scheduling jobs with similar processing times on multiprocessor systems using LPT-algorithm. *Computing*, Vol.46, 183-191.

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

Li, R. & Huang, H. C. (2004) On-line Scheduling for Jobs with Arbitrary Release Times. *Computing*, Vol.73, 79-97.

Li, R. & Huang, H. C. (2007) Improved Algorithm for a Generalized On-line Scheduling Problem. *European Journal of operational research*, Vol.176, 643-652.

Liu, W. P., Sidney, J. B. & Vliet, A. (1996) Ordinal algorithm for parallel machine scheduling. *Operations Research Letters*. Vol.18, 223-232.

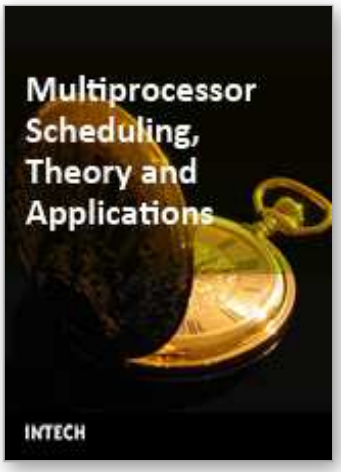
Motwani, R., Phillips, S. & Torng, E. (1994) Non-clairvoyant scheduling. *Theoretical computer science*, Vol.130, 17-47.

Seiden, S., Sgall, J., & Woeginger, G. J. (2000) Semi-online scheduling with decreasing job sizes. *Operations Research Letters*. Vol.27, 215-221.

Shmoys, D. B., Wein, J. & Williamson, D. P. (1995) Scheduling parallel machines on-line. *SIAM. J. Computing*, Vol.24, 1313-1331.

Tan, Z. Y. & He, Y. (2001) Semi-online scheduling with ordinal data on two Uniform Machines. *Operations Research Letters*. Vol.28, 221-231.

Tan, Z. Y. & He, Y. (2002) Semi-online problem on two identical machines with combined partial information. *Operations Research Letters*. Vol.30, 408-414.



Multiprocessor Scheduling, Theory and Applications

Edited by Eugene Levner

ISBN 978-3-902613-02-8
Hard cover, 436 pages
Publisher I-Tech Education and Publishing
Published online 01, December, 2007
Published in print edition December, 2007

A major goal of the book is to continue a good tradition - to bring together reputable researchers from different countries in order to provide a comprehensive coverage of advanced and modern topics in scheduling not yet reflected by other books. The virtual consortium of the authors has been created by using electronic exchanges; it comprises 50 authors from 18 different countries who have submitted 23 contributions to this collective product. In this sense, the volume can be added to a bookshelf with similar collective publications in scheduling, started by Coffman (1976) and successfully continued by Chretienne et al. (1995), Gutin and Punnen (2002), and Leung (2004). This volume contains four major parts that cover the following directions: the state of the art in theory and algorithms for classical and non-standard scheduling problems; new exact optimization algorithms, approximation algorithms with performance guarantees, heuristics and metaheuristics; novel models and approaches to scheduling; and, last but least, several real-life applications and case studies.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821