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

Optimal online algorithms for scheduling on two identical machines under a grade of service*

JIANG Yi-wei (蒋义伟)†1,2, HE Yong (何 勇)2, TANG Chun-mei (唐春梅)2

(1*Laboratory of Information and Optimization Technologies, Ningbo Institute of Technology, Zhejiang University, Ningbo 315100, China*)

(2*Department of Mathematics, Zhejiang University, Hangzhou 310027, China*)

†E-mail: jywzju@163.com

Received Feb. 10, 2005; revision accepted Apr. 19, 2005

Abstract: This work is aimed at investigating the online scheduling problem on two parallel and identical machines with a new feature that service requests from various customers are entitled to many different grade of service (GoS) levels, so each job and machine are labelled with the GoS levels, and each job can be processed by a particular machine only when its GoS level is no less than that of the machine. The goal is to minimize the makespan. For non-preemptive version, we propose an optimal online algorithm with competitive ratio 5/3. For preemptive version, we propose an optimal online algorithm with competitive ratio 3/2.

Key words: Online algorithm, Competitive analysis, Parallel machine scheduling, Grade of service (GoS)
doi:10.1631/jzus.2006.A0309 **Document code:** A **CLC number:** TP393; O223

INTRODUCTION

We study the problem of online scheduling on two identical parallel machines with a new feature that service requests from various customers are entitled to many different grade of service (GoS) levels. The goal is to minimize the makespan under the constraint that all requests are satisfied. This problem was first proposed by Hwang *et al.*(2004) and is motivated by the following scenario. In the service industry, the service providers often have special customers, such as, gold, silver, or platinum members who are more valued than the regular members. Those special members are usually entitled to premium services, so some kind of special service policy must be implemented by the service provider. One simple scheme for providing special service is to label machines and jobs with pre-specified GoS levels and allow each job to be processed by a particular machine only when its GoS level is no less than that of the machine. In effect, the processing capability of the machines labelled with high GoS levels tends to be reserved for jobs with high GoS levels. In such situation, assigning jobs to the machines becomes a parallel machine scheduling problem with a special eligibility constraint.

Formally, this problem can be described as follows. We are given a sequence $J=\{p_1, p_2, \ldots, p_n\}$ of independent jobs with positive processing times, which must be processed on m identical machines M_1, M_2, …, M_m. We identify jobs with their processing times. Each job p_j is labelled with the GoS level of $g(p_j)$, and each machine M_i is also labelled with the GoS level of $g(M_i)$. M_i is allowed to process job p_j only if $g(M_i)\leq g(p_j)$. So, if a job has a GoS level less than both levels of all the machines, then the job has to be rejected. Without loss of generality, we assume no such job exists in this paper. The objective is to minimize the makespan, i.e., the maximum comple-

*Project supported by the National Natural Science Foundation of China (No. 10271110) and the Teaching and Research Award Program for Outstanding Young Teachers in Higher Education, Institutions of MOE, China tion time of all jobs. This problem is called parallel machine scheduling with GoS eligibility (Hwang *et al*., 2004).

We consider online algorithms in this paper, so we assume that jobs arrive in online over list. That is to say, jobs arrive one by one and are required to be scheduled irrevocably on machines as soon as they are given, without any knowledge of the jobs that will arrive later. If we have full information on the job data before constructing a schedule, it is called offline. Algorithms for an online/offline problem are called online/offline algorithms.

Both non-preemptive and preemptive algorithms are considered in this paper. For a preemptive algorithm, a job may be cut into a few pieces for assignment to possibly different machines, in non-overlapping time slots. For a non-preemptive algorithm, jobs are not allowed to be cut and have to be processed continuously on a machine. Note that in online algorithm design, we may get benefit to introduce idle time between two consecutive jobs on the same machine and fill it for the subsequent jobs, especially for preemptive algorithms, see for example, references (Epstein and Favrholdt, 2002; He and Jiang, 2004). But it may violate a basic service rule of first-come-first serve for jobs assigned to the same machine, so, in this paper we assume that idle time is not allowed to be introduced before the last job is completed.

The performance of an online algorithm is measured by its competitive ratio. For a job sequence J and an algorithm A, let $c^A(J)$ (or in short c^A) denote the makespan produced by A and let $c^*(J)$ (or in short c^*) denote the optimal makespan in an offline version. Then the competitive ratio of A is defined as $R_A=\sup\{c^A(J)/c^*(J)\}$. An online problem has a lower bound ρ if no online deterministic algorithm has a competitive ratio smaller than ρ. An online algorithm is called optimal if its competitive ratio matches the lower bound.

Clearly, the offline version of this problem is NP-hard. Note that Lenstra *et al*.(1990) proposed a binary search algorithm based on linear programming with makespan no more than 2 times the optimum for the most general problem of unrelated parallel machine scheduling, which certainly covers the problem under consideration. Recently, for the problem under consideration, Hwang *et al*.(2004) presented an offline algorithm LG-LPT with makespan no more than 5/4-times the optimum for m=2, and $(2-1/(m-1))$-times the optimum for $m\geq3$. For the online version, Azar *et al*.(1995) presented an online algorithm with competitive ratio $\log_2 2m$ for any m. In particular, the competitive ratio turns into 2 for m=2. However, all the results are for non-preemptive algorithms, and no result is obtained in the literature for preemptive algorithms of the considered problem.

In this paper, we will consider the online version of this problem on two identical machines. Without loss of generality, we assume that $g(M_i)=i$, i=1,2. Since if both machines have the same GoS level, the problem becomes the classical parallel machine scheduling aimed at minimizing the makespan, which has been extensively studied in the literature. We will propose an optimal non-preemptive online algorithm with competitive ratio 5/3, which greatly improves the known upper bound 2 (Azar *et al*., 1995), and will consider the preemptive case and present an optimal preemptive online algorithm with competitive ratio 3/2.

Note that as pointed out in (Hwang *et al*., 2004), the offline version of two identical machine cases is equivalent to parallel machine scheduling with non-simultaneous machine available time (Lee, 1991; Lin *et al*., 1997; Lee *et al*., 2000). However, for the online version, these two problems are definitely different, and an online algorithm for one problem cannot be applied to another directly.

The rest of the paper is organized as follows. Section 2 gives some basic notations and lower bounds for the problem. Section 3 presents an optimal non-preemptive online algorithm. Finally, Section 4 presents an optimal preemptive online algorithm.

PRELIMINARY

To simplify the presentation, we use the following notations in the remainder of the paper. Define moment $j\geq0$ as the moment right after the jth job is scheduled. Denote $T_j=\sum_{i=1}^{j}p_i$ and $p_j^{\max}=\max\{p_k \mid k=1,\ldots,j\}$. Let L_j^i denote the completion time of machine M_i at moment j in an online algorithm A, i=1,2. Let L_j^A and L_j^* be the current makespan yielded by algorithm A and the optimal makespan at moment j, respectively. Then $c^A=L_n^A$ and $c^*=L_n^*$. Let $P_1=\{p_k|g(p_k)=1,k=1,2,\ldots,n\}$, $P_2=\{p_k|g(p_k)=2,k=1,2,\ldots,n\}$, and $S_j=\sum_{p_i\in P_1,i\le j}p_i$.

Lemma 1 No matter whether preemption is allowed or not, the optimal makespan is at least $LB_j\equiv\max\{p_j^{\max},T_j/2,S_j\}$ at any moment $j\geq1$.

Proof It is clear that the optimal makespan satisfies $L_j^*\geq\max\{p_j^{\max},T_j/2\}$ at any moment j. By the definition of the problem, all the jobs in set P_1 can only be processed on machine M_1, which implies that the optimal makespan is not smaller than the current total size of the jobs in P_1, that is, $L_j^*\geq S_j$ according to the definition of S_j. The proof is completed.

Theorem 1 The competitive ratio of any online non-preemptive algorithm is at least 5/3.

Proof We use adversary method to establish the result. Let A be an non-preemptive online algorithm with competitive ratio C. The first two jobs with $p_1=p_2=1$ and $g(p_1)=g(p_2)=2$ arrive. If both of them are assigned to one machine by algorithm A, then no more job arrives. It follows that $c^A=2$ and $c^*=1$ which imply that $C\geq2$. Therefore, we assume that the algorithm schedules the two jobs on different machines. Then the third job with $p_3=1$ and $g(p_3)=2$ arrives. If it is scheduled on machine M_1, then the last job with $p_4=3$ and $g(p_4)=1$ arrives. Therefore we have $c^A\geq5$ and $c^*=3$, from which it follows that $C\geq5/3$. On the other hand, if the third job is scheduled on machine M_2, then the fourth job with $p_4=3$ and $g(p_4)=2$ arrives. If p_4 is assigned to M_2, then we have $c^A=5$ and $c^*=3$, from which it follows that $C\geq5/3$. If the fourth job is assigned to M_1, we have $L_4^1=4$ and $L_4^2=2$. Then the last job with $p_5=6$ and $g(p_5)=1$ arrives. It follows that $c^A=10$ and $c^*=6$, implying $C\geq5/3$.

In summary, we have shown that any online algorithm has a competitive ratio of at least $C\geq5/3$.

Theorem 2 The competitive ratio of any preemptive online algorithm is at least 3/2.

Proof Similarly, let A be an online preemptive algorithm with competitive ratio C. The first job with $p_1=1$ and $g(p_1)=2$ arrives. p_1 must be assigned completely to a machine (Splitting it would introduce idle time, which is not allowed in our problem). If it is assigned to machine M_1, then the second and last job with $p_2=1$ and $g(p_2)=1$ arrives. Since p_2 has to be assigned to machine M_1 as $g(p_2)=1$, it follows that the makespan of algorithm A is 2, while the optimal makespan is 1. So, we have $C\geq2>3/2$. On the other hand, if p_1 is assigned to machine M_2, then when the second job with $p_2=1$ and $g(p_2)=1$ arrives, it has to be assigned to machine M_1. Now both loads of the two machines are 1. Then the third and last job with $p_3=2$ and $g(p_3)=2$ arrives. We thus have that the makespan of algorithm A is at least 3, while the optimal makespan is 2. Hence, we obtain $C\geq3/2$, too.

AN OPTIAMAL NON-PREEMPTIVE ALGORITHM

In this section, we present an optimal non-preemptive algorithm for the considered problem, which can be formally described as follows.

Algorithm *H*1:

```
0. Let $L_1^0=L_2^0=0$ and $j=0$.
1. While job $p_j$ exists, do {
2. If $p_j\in P_1$, schedule $p_j$ on machine $M_1$.
3. Else {define $a=p_j+L_{j-1}^2-5L_{j-1}^1$ and
              $b=p_j+L_{j-1}^1-2L_{j-1}^2$.
     (i) if $a>0$ and $b\leq0$, schedule $p_j$ on machine $M_1$.
     (ii) else schedule $p_j$ on machine $M_2$.}
4. Let $j=j+1$.}
```

Lemma 2 If $L_j^2/5\leq L_j^1\leq5L_j^2$ at moment j, then we have $L_j^{H1}/L_j^*\leq5/3$.

Proof $L_j^2/5\leq L_j^1\leq5L_j^2$ implies that $L_j^1\leq5(L_j^1+L_j^2)/6$ and $L_j^2\leq5(L_j^1+L_j^2)/6$. Hence, $L_j^{H1}\leq\max\{L_j^1,L_j^2\}\leq5(L_j^1+L_j^2)/6=5T_j/6$. On the other hand, we have $L_j^*\geq\max\{p_j^{\max},T_j/2,S_j\}\geq T_j/2$ due to Lemma 1. Combining them, we obtain that $\frac{L_j^{H1}}{L_j^*}\leq\frac{5T_j/6}{T_j/2}\leq\frac{5}{3}$.

Lemma 3 If $L_j^1>5L_j^2$ at moment j, then we have: (a) $S_j>T_j/2$; and (b) $L_j^{H1}/L_j^*\leq5/3$.

Proof (a) Let p_k, $k\leq j$, be the last job such that $L_k^1>2L_k^2$ and $L_{k-1}^1\leq2L_{k-1}^2$. That is to say, at any moment $k\leq i\leq j$, $L_i^1>2L_i^2$ holds. Then we can conclude that $p_k \in P_1$ by the algorithm rule.

We claim that, in $\{p_{k+1}, \ldots, p_j\}$, all those jobs assigned to machine M_1 are from set P_1. To see it, suppose that there is a job $p_i \in P_2$, $k<i\leq j$, which is assigned to machine M_1. It implies that p_i is scheduled by Step 3(i) of Algorithm $H1$. So, we know that the current value of b satisfies $b\leq 0$, that is, $L_i^1 \leq 2L_i^2$, which contradicts the definition of p_k. Hence, the claim is true. It implies that

$$S_j \geq L_j^1 - L_{k-1}^1. \tag{1}$$

From $L_j^1 > 5L_j^2$ and $T_j = L_j^1 + L_j^2$, we obtain that $L_j^1 > 5T_j/6$ and $L_j^2 < T_j/6$. And from $L_{k-1}^1 \leq 2L_{k-1}^2$, we have $L_{k-1}^1 \leq 2L_{k-1}^2 \leq 2L_j^2 < T_j/3$. Combining it with Eq.(1), we have $S_j > 5T_j/6 - T_j/3 = T_j/2$.

(b) Lemma 1 implies $L_j^* \geq \max\{p_j^{\max}, T_j/2, S_j\} \geq S_j$. Moreover, by $L_j^1 > 5L_j^2$ and Eq.(1), we have $L_j^{H1} = L_j^1 \leq S_j + L_{k-1}^1$. Therefore, we have

$$\frac{L_j^{H1}}{L_j^*} \leq \frac{S_j + L_{k-1}^1}{S_j} = 1 + \frac{L_{k-1}^1}{S_j} < 1 + \frac{T_j/3}{T_j/2} \leq \frac{5}{3}.$$

Lemma 4 If $a>0$ and $b>0$ at moment j, then we have: (a) $p_j>T_{j-1}$, that is, $p_j = p_j^{\max}$ and $p_j>T_j/2$; and (b) $L_j^{H1}/L_j^* \leq 5/3$.

Proof (a) By the assumption and definitions of a and b, we have

$$p_j > 5L_{j-1}^1 - L_{j-1}^2 \tag{2}$$

and

$$p_j > 2L_{j-1}^2 - L_{j-1}^1 \tag{3}$$

at moment j. Suppose $p_j \leq T_{j-1} = L_{j-1}^1 + L_{j-1}^2$. Then according to Eq.(2), we have $5L_{j-1}^1 - L_{j-1}^2 < L_{j-1}^1 + L_{j-1}^2$, i.e., $L_{j-1}^2 > 2L_{j-1}^1$. On the other hand, from Eq.(3), we obtain that $2L_{j-1}^2 - L_{j-1}^1 < L_{j-1}^1 + L_{j-1}^2$, i.e., $L_{j-1}^2 < L_{j-1}^1$. It is a contradiction. Thus we conclude that $p_j > T_{j-1}$, and $p_j = p_j^{\max}$ obviously.

(b) From Lemma 1, we have $L_j^* \geq \max\{p_j^{\max}, T_j/2, S_j\} \geq p_j^{\max} = p_j$. From (a), we have $L_j^{H1} = L_j^2 \leq p_j + L_{j-1}^2$. From Eq.(3), we have $2L_j^2 < p_j + L_{j-1}^1 = T_j - L_{j-1}^2$, resulting in $L_{j-1}^2 < T_j/3$. Thus we have

$$\frac{L_j^{H1}}{L_j^*} \leq \frac{p_j + L_{j-1}^2}{p_j} = 1 + \frac{L_{j-1}^2}{p_j} < 1 + \frac{T_j/3}{T_j/2} \leq \frac{5}{3}.$$

Theorem 3 The competitive ratio of Algorithm $H1$ is 5/3. Thus it is optimal.

Proof We show that $L_j^{H1}/L_j^* \leq 5/3$ holds for every j=1,2,…,n by induction method. The result is trivially true at moment 1. Assume $L_{j-1}^{H1}/L_{j-1}^* \leq 5/3$ holds at moment j–1 ($j\geq 2$). Now we consider p_j ($j\geq 2$). Two cases are considered according to the value of $g(p_j)$.

Case 1 $g(p_j)$=1. By the algorithm rule, p_j is assigned to machine M_1.

(a) If $L_j^2/5 \leq L_j^1 \leq 5L_j^2$, then we have $L_j^{H1}/L_j^* \leq 5/3$ due to Lemma 2.

(b) If $L_j^1 < L_j^2/5$, since p_j is assigned to machine M_1, we know that $L_j^2 = L_{j-1}^2$. Hence, $L_j^{H1} = L_j^2 = L_{j-1}^2 = L_{j-1}^{H1}$, that is, the current makespan yielded by $H1$ is unchanged after assigning job p_j. Because $L_j^* \geq L_{j-1}^*$, we obtain $L_j^{H1}/L_j^* \leq L_{j-1}^{H1}/L_{j-1}^* \leq 5/3$ by induction.

(c) If $L_j^1 > 5L_j^2$, we obtain the result directly from Lemma 3.

Case 2 $g(p_j)$=2. We distinguish three subcases according to the assignment of p_j.

Subcase 2.1 $a\leq 0$. It implies that p_j is scheduled on machine M_2 and $L_j^2 \leq 5L_j^1$, i.e., $L_j^1 \geq L_j^2/5$. Moreover, if $L_j^1 \leq 5L_j^2$, we can obtain $L_j^{H1}/L_j^* \leq 5/3$ due to Lemma 2. If $L_j^1 > 5L_j^2$, since p_j is assigned to machine M_2, we can conclude that the current makespan yielded by $H1$ is unchanged after assigning job p_j, that is, $L_j^{H1} = L_j^1 = L_{j-1}^1 = L_{j-1}^{H1}$, from which it follows that $L_j^{H1}/L_j^* \leq L_{j-1}^{H1}/L_{j-1}^* \leq 5/3$ by induction.

Subcase 2.2 $a>0$ and $b\leq 0$. It implies that p_j is scheduled on machine M_1 and $L_j^1 \leq 2L_j^2$. Moreover, if $L_j^1 \geq L_j^2/5$, we can obtain $L_j^{H1}/L_j^* \leq 5/3$ due to Lemma 2. If $L_j^1 < L_j^2/5$, we have $L_j^{H1}/L_j^* \leq 5/3$ by the same argument for (b) of Case 1.

Subcase 2.3 $a>0$ and $b>0$. It is easy to obtain $L_j^{H1}/L_j^* \leq 5/3$ by Lemma 4.

We have thus proved that $L_j^{H1}/L_j^* \leq 5/3$. Moreover, Algorithm $H1$ is optimal by Theorem 1.

AN OPTIAMAL PREEMPTIVE ALGORITHM

In this section, we present an optimal preemptive algorithm for the considered problem, which can be formally described as follows.

Algorithm *H*2:

```
0. Let $L_1^0 = L_2^0 = 0$ and $j=0$.
1. While job $p_j$ exists, do {
2. If $p_j \in P_1$, schedule $p_j$ on machine $M_1$.
3. Else {
    compute the value of $LB_j$ according to Lemma 1 .
    (i) if $L_{j-1}^2 + p_j \leq \frac{3}{2}LB_j$, schedule $p_j$ on machine
        $M_2$ completely.
    (ii) else schedule the part $\frac{3}{2}LB_j - L_{j-1}^2$ of $p_j$ on
        machine $M_2$ and the leftover on machine $M_1$.
    }
4. Let $j=j+1$.}
```

Clearly, to show the feasibility of Algorithm $H2$, we only need to prove that the assignment of the job p_j scheduled by Step 3(ii) is feasible, that is, the time slots assigned to p_j on two machines do not overlap. Furthermore, to obtain that the competitive ratio of $H2$ is 3/2, it suffices to verify $L_j^{H2}/L_j^* \leq 3/2$ for $p_j \in P_1$ since the assignment of $p_j \in P_2$ satisfies $L_j^{H2}/L_j^* \leq 3/2$ obviously (because of the algorithm rule of Step 3). The detailed arguments begin with the following lemma.

Lemma 5 Algorithm $H2$ is feasible.

Proof As stated above, we only have to show the time slots assigned to p_j in Step 3(ii) do not overlap, which is equivalent to showing

$$L_j^1 = L_{j-1}^1 + p_j - \left(\frac{3}{2}LB_j - L_{j-1}^2\right) \leq L_{j-1}^2, \tag{4}$$

i.e.,

$$L_{j-1}^1 + p_j \leq \frac{3}{2}LB_j. \tag{5}$$

We prove Eq.(5) by contradiction. Suppose

$$L_{j-1}^1 + p_j > \frac{3}{2}LB_j. \tag{6}$$

Note that the algorithm rule in Step 3(ii) implies

$$L_{j-1}^2 + p_j > \frac{3}{2}LB_j. \tag{7}$$

By Eqs.(6), (7) and Lemma 1, we have $L_{j-1}^1 + p_j + L_{j-1}^2 + p_j > 3LB_j \geq 3T_j/2$. Since $T_j = T_{j-1} + p_j = L_{j-1}^1 + L_{j-1}^2 + p_j$, we obtain

$$p_j > T_j/2. \tag{8}$$

It follows that $p_j = p_j^{\max}$, and $LB_j \geq p_j$ by Lemma 1. Then we have $L_{j-1}^1 + p_j > 3p_j/2$ and $L_{j-1}^2 + p_j > 3p_j/2$ from Eqs.(6) and (7). Combining them with $T_j = T_{j-1} + p_j = L_{j-1}^1 + L_{j-1}^2 + p_j$, we obtain that $p_j < T_j/2$, which contradicts Eq.(8). Thus Eq.(5) holds. The proof is completed.

Lemma 6 Suppose that there exists at least one job in P_2 for which a part of this job is processed on machine M_1, and let p_l be the last of such a job. Then we have $L_l^1 \leq L_l^2$.

Proof From the algorithm description, we can see that the job p_l must be processed by Step 3(ii). With an argument analogous to the proof of Eq.(4), we can obtain that $L_l^1 \leq L_{l-1}^2$. As $L_{l-1}^2 \leq L_l^2$ holds trivially, we have $L_l^1 \leq L_l^2$.

Theorem 4 The competitive ratio of Algorithm $H2$ is 3/2. So it is optimal.

Proof By Lemma 1 we have $L_j^* \geq LB_j$. Therefore, to obtain $L_j^{H2}/L_j^* \leq 3/2$, it suffices to show $L_j^{H2}/LB_j \leq 3/2$. We distinguish two cases according to the grade of the job achieving the final makespan of Algorithm *H*2.

Case 3 The makespan of Algorithm *H*2 is determined by a job from P_2. It is not hard to obtain that $L_j^{H2}=L_j^2$ by the assignment of jobs in P_2. From the algorithm description in Steps 3(i) and 3(ii) we have $L_j^{H2}/LB_j\le 3/2$.

Case 4 The makespan of Algorithm *H*2 is determined by a job from P_1. As we know that the jobs from P_1 must be processed on machine M_1, we have $L_j^{H2}=L_j^1$.

If there does not exist a job from P_2 processed on machine M_1, that is, all the jobs processed on machine M_1 belong to P_1, then we can conclude that $L_j^{H2}=S_j\le LB_j$ by Lemma 1. If there exist some jobs from P_2 processed on machine M_1, we denote $S=L_j^1-L_l^1$, where l is defined in the same way as in Lemma 6. Then we have $S_j\ge S$, and $L_l^1\le L_l^2\le L_j^2$ by Lemma 6.

Now we are ready to prove $L_j^{H2}/LB_j\le 3/2$ for Case 4.

If $S\le L_l^1+L_j^2$, combining $L_l^1\le L_j^2$ and $LB_j\ge\frac{T_j}{2}=\frac{S+L_l^1+L_j^2}{2}$, we have

$$\frac{L_j^{H2}}{LB_j}\le\frac{2(S+L_l^1)}{S+L_l^1+L_j^2}=2-\frac{2L_j^2}{S+L_l^1+L_j^2}$$
$$\le 2-\frac{2L_j^2}{2(L_l^1+L_j^2)}\le 2-\frac{L_j^2}{L_j^1+L_j^2}=\frac{3}{2}.$$

If $S>L_l^1+L_j^2$, i.e., $2S>S+L_l^1+L_j^2=T_j$, then it follows that $S_j\ge S>T_j/2$. Hence, we have $LB_j\ge S_j$ by Lemma 1. Since $L_l^1\le L_j^2$, we have $L_l^1<S-L_j^2\le S-L_l^1$, i.e., $L_l^1<S/2$. Hence

$$\frac{L_j^{H2}}{LB_j}\le\frac{S+L_l^1}{S_j}\le\frac{S+S/2}{S}=\frac{3}{2}.$$

By now we have completed the proof of the competitive ratio. Moreover, the optimality of Algorithm *H*2 is a direct consequence of Theorem 2.

References

Azar, Y., Naor, J., Rom, R., 1995. The competitiveness of on-line assignments. *Journal of Algorithms*, **18**(2): 221-237. [doi:10.1006/jagm.1995.1008]

Epstein, L., Favrholdt, L., 2002. Optimal preemptive semi-online scheduling to minimize makespan on two related machines. *Operations Research Letters*, **30**(4): 269-275. [doi:10.1016/S0167-6377(02)00179-7]

He, Y., Jiang, Y.W., 2004. Optimal algorithms for semi-online preemptive scheduling problems on two uniform machines. *Acta Informatica*, **40**(5):367-383. [doi:10.1007/s00236-003-0134-7]

Hwang, H., Chang, S., Lee, K., 2004. Parallel machine scheduling under a grade of service provision. *Computers & Operations Research*, **31**(12):2055-2061. [doi:10.1016/S0305-0548(03)00164-3]

Lee, C.Y., 1991. Parallel machine scheduling with non-simultaneous machine available time. *Discrete Applied Mathematics*, **30**(1):53-61. [doi:10.1016/0166-218X(91)90013-M]

Lee, C.Y., He, Y., Tang, G.C., 2000. A note on "Parallel machine scheduling with non-simultaneous machine available time". *Discrete Applied Mathematics*, **100**(1-2):133-135. [doi:10.1016/S0166-218X(99)00201-2]

Lenstra, J.K., Shmoys, D.B, Tardos, N.E., 1990. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming*, **46**(1-3):259-271. [doi:10.1007/BF01585745]

Lin, G., He, Y., Yao, Y., Lu, H., 1997. Exact bounds of the modified LPT algorithms applying to parallel machines scheduling with non-simultaneous machine available times. *Applied Math.-JCU*, **12B**:109-116.