Semi-online scheduling with “end of sequence” information[*]

Leah Epstein[†] Deshi Ye[‡§]

Abstract

We study a variant of classical scheduling, which is called *scheduling with “end of sequence” information*. It is known in advance that the last job has the longest processing time. Moreover, the last job is marked, and thus it is known for every new job whether it is the final job of the sequence. We explore this model on two uniformly related machines, that is, two machines with possibly different speeds. Two objectives are considered, maximizing the minimum completion time and minimizing the maximum completion time (makespan). Let s be the speed ratio between the two machines, we consider the competitive ratios which are possible to achieve for the two problems as functions of s. We present algorithms for different values of s and lower bounds on the competitive ratio. The proposed algorithms are best possible for a wide range of values of s. For the overall competitive ratio, we show tight bounds of $\phi + 1 \approx 2.618$ for the first problem, and upper and lower bounds of 1.5 and 1.46557 for the second problem.

1 Introduction

The traditional model of online scheduling assumes that the length of the job stream is not known in advance, and the algorithm needs to obey the competitive ratio at every time. This reflects the structure of some systems. However, it is often the case that the worst case occurs when the largest job arrives last (see e.g. the seminal paper of Graham [8]). In real world systems, even though the information on future jobs is not complete, it is still known for each job whether the current job is last or not.

In this paper we try to model this situation and study the following relatively new semi-online scheduling problem on two uniformly related machines. In this problem it is known that the very last job has the longest processing time (though previous jobs in the sequence may have the same processing time as well), and the adversary who presents the sequence informs the scheduler upon arrival of each job not only the size of the job, but also whether it is the last job.

In the uniformly related machines system model, each machine M_i has a *speed* s_i and each job J_k has an initial processing time (or size) p_k. The processing time of job J_k on machine M_i is p_k/s_i. Jobs arrive one

[*]The authors would like to dedicate this paper to the memory of our colleague and friend **Yong He** who passed away in August 2005 after struggling with illness.

[†]Department of Mathematics, University of Haifa, 31905 Haifa, Israel. lea@math.haifa.ac.il.

[‡]Corresponding author.

[§]College of Computer Science, Zhejiang University, Hangzhou 310027, China. Research was supported in part by NSFC(10601048). yedeshi@zju.edu.cn.

by one and are are presented to a scheduler in this way. Each job must be assigned to one of the machines before any future jobs are revealed. No preemption is allowed. We study two goals which are minimizing the makespan and maximizing the minimum completion time. In our model, which is stated above, we get additional information in advance compared to the standard online model. Therefore we see this problem as a semi-online scheduling problem.

We use the *competitive ratio* to measure the quality of a semi-online algorithm, similarly to online algorithms. For any input job sequence I, let $C_A(I)$ and $C^*(I)$ denote the values of the solutions of an algorithm A and optimal offline algorithm which knows the whole sequence in advance, respectively. For the objective to minimize the makespan, we say that R_A is the competitive ratio of the algorithm A if

$$R_A = \sup_I \left\{ \frac{C_A(I)}{C^*(I)} \right\}.$$

On the other hand, for the objective to maximize the minimum completion time, the competitive ratio R_A of the algorithm A is defined as

$$R_A = \sup_I \left\{ \frac{C^*(I)}{C_A(I)} \right\}.$$

For simplicity, we use C_A and C^* instead of $C_A(I)$ and $C^*(I)$ if the meaning is clear from the context.

Previous Work: The semi-online scheduling model studied in this paper was first suggested by Zhang and Ye [17]. They study the standard makespan minimization problem on two and three identical machines (i.e., machines that all have the same speed). They provided optimal algorithms with competitive ratios $\sqrt{2}$ and $3/2$ for two and three machines respectively. They also pointed out that all the additional information given by an adversary is necessary for the model. That is, the model is no different from the standard model without having all the extra information (it is not enough that the last job is largest if is not announced to be last, and it is not enough to announce the last job if it is not largest).

Various other models of semi-online scheduling problems on identical machines have been studied extensively in recent years [11, 16, 10, 13, 2, 6, 15, 3, 12]. These variants differ in which partial information about future job is known in advance. A few examples are listed below.

In the first model jobs arrive in non-increasing order of processing time [9, 13]. Several models assume some information is known in advance, this can be the total processing time of jobs [11, 1], the optimal makespan [2, 4] or the largest processing time [10]. A model where all job processing times are bounded between two known values, with a ratio of at most ρ for some given $\rho > 1$ between these two values, was studied in [10]. On the other hand, some semi-online problems give the algorithm more power than a general online problem, and jobs do not always need to be assigned right away. One such model is studied in [11, 16], where a buffer is available for storage of a small number of jobs. A semi-online algorithm in [12] benefits from allowing the current assignment to be changed whenever a new job arrives, subject to the constraint that the total processing time of moved jobs is bounded by β times the processing time of the arriving job.

Our Contributions: In this paper we focus on the two uniformly related machines case. Two objective functions of the problem are explored, (maximization of the) smallest completion time of any machine (Max-min) and (minimization of) the maximum completion time of any machine, also called the makespan (Min-max).

The processing time of a job on a given machine is also called the workload of the job on that machine. The workload of a machine is the sum of workloads of all jobs assigned to it. Thus, the makespan is the maximum workload among all the machines and the minimum completion time is the minimum workload of any machine. For two identical machines ($s = 1$), the optimal competitive ratio for the Min-max problem is $\sqrt{2}$ [17]. We show that for the Max-min problem this value equals to $1 + \frac{\sqrt{2}}{2}$.

We present algorithms relying on the speed ratio s of machines for both problems. The competitive ratios depend also on the speed s. The proposed algorithms are best possible for almost all the value of s. Table 1 and Table 2 list the results (see also Figures 1 and 2), where

$$R(s) = \frac{3s + 1 + \sqrt{5s^2 + 2s + 1}}{2s + 2} ,$$

$$R'(s) = \frac{s - 1 + \sqrt{5s^2 + 2s + 1}}{2s} .$$

Table 1: Bounds on two uniformly related machines, Max-min

speed s	*Lower bound*	*Upper bound*
$s \geq 8.24264$	$R(s)$	$R(s)$
$7.159191247 < s < 4 + 3\sqrt{2} = 8.24264$	$R_1(s)$	$R(s)$
$5.40431 < s < 7.159191$	$R(s_1) \approx 2.324718$	$R(s)$
$1 \leq s \leq s_1 \approx 5.40431$	$R(s)$	$R(s)$

Some exact values in Table 1 are as follows. $R_1(s) = (\frac{1}{27} + \frac{s}{2} + \frac{\sqrt{12s+81s^2}}{18})^{\frac{1}{3}} + (\frac{1}{27} + \frac{s}{2} - \frac{\sqrt{12s+81s^2}}{18})^{\frac{1}{3}} + \frac{1}{3}$, $s_1 = \frac{5}{3} + \left(\frac{367}{54} - \frac{5\sqrt{69}}{18}\right)^{\frac{1}{3}} + \left(\frac{367}{54} + \frac{5\sqrt{69}}{18}\right)^{\frac{1}{3}}$ and $R(s_1) = (\frac{1}{2} + \frac{\sqrt{69}}{18})^{\frac{1}{3}} + (\frac{1}{2} - \frac{\sqrt{69}}{18})^{\frac{1}{3}} + 1$.

Table 2: Bounds on two uniformly related machines, Min-max

speed s	*Lower bound*	*Upper bound*
$s \geq 1 + \sqrt{3}$	$1 + \frac{1}{s}$	$1 + \frac{1}{s}$
$2 \leq s < 1 + \sqrt{3}$	$1 + \frac{2s}{2+2s+s^2}$	$1 + \frac{1}{s}$
$1.46557 \leq s < 2$	$1 + \frac{s}{s^2+1}$	$R'(s)$
$1 < s < s_4 = 1.46557$	$R'(s)$	$R'(s)$
$s = 1$	$\sqrt{2}$	$\sqrt{2}$

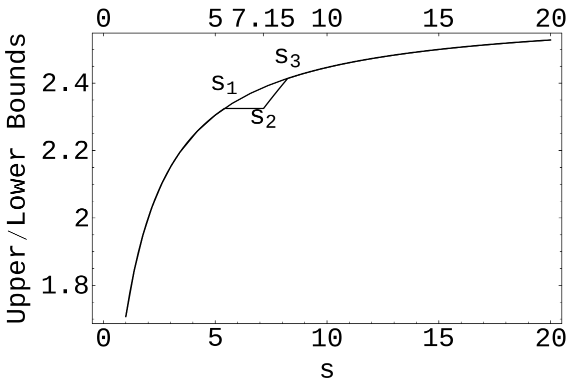

Figure 1: Upper bound and lower bound as a function of s, where $s_1 = 5.40431$, $s_2 = 7.15191247$, $s_3 = 8.24264$. Max-min.

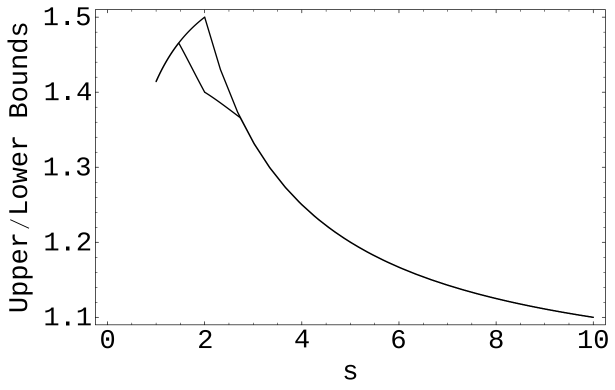

Figure 2: Upper bound and lower bound as a function of s. Min-max.

The exact value of s_4 in Table 2 is $s_4 = \frac{1}{6}(116+12\sqrt{93})^{1/3} + \frac{1}{6}(116-12\sqrt{93})^{1/3} + \frac{1}{3}$. Note that this implies upper and lower bounds of $\phi + 1 \approx 2.618$ for the Max-min problem. For the Min-max problem we get an upper bound of 1.5 and a lower bound of 1.46557.

The online versions of the problems studied in this paper were studied in [7] (Min-max) and [5] (Max-min). For Min-max model, in [7], it was shown that the best competitive ratio for minimizing makespan is $\min\{1+\frac{1}{s}, 1+\frac{s}{s+1}\}$. Comparing it to our results we can see that for small s our semi-online model indeed improves the situation of the algorithm. However, for large enough s, we see that the situation does not change. Note that for these values of s the competitive ratio is already quite small. For Max-min model, in [5], it was shown that for the basic online model, the best competitive ratio is $s+1$. For this problem the additional information we have is helpful especially for large values of s where we can show a constant competitive ratio unlike the case for the online problem.

The rest of the paper is organized as follows. In Section 2 we define a general algorithm. In Section 3 we study the problem of maximizing the minimum completion time. In Section 4 we investigate the problem of minimizing the makespan.

2 Algorithms

We start with defining a general algorithm which will be used both in Section 3 and in Section 4. The idea of the algorithm is to keep the faster machine relatively empty, and to assign the very last job to this machine. Clearly, for some inputs this machine cannot be kept completely empty, and we fill it with some amount of jobs.

For the definition and analysis of algorithms for our problems, we introduce some notations. We denote jobs and their processing times by J_t and p_t ($t = 0, 1, 2, \ldots$), respectively. Denote by $1/s$ ($s \geq 1$) the speed of the slow machine, without loss of generality, the speed of fast machine is 1. Machine 1 (M_1) is the slow machine and Machine 2 (M_2) is the fast machine. Let $M_i(t)$ denote the workload of Machine i immediately after the job J_t is assigned, for $t = 0, 1, 2, \ldots$, and $i = 1, 2$. Let $\alpha(s)$ be a function of s such that for any value of s, $0 < \alpha(s) < \frac{1}{s}$. We define $\beta(s) = \frac{\alpha(s)}{1 - s\alpha(s)}$. According to the constraints on $\alpha(s)$, $\beta(s)$ is defined for every value of s and is positive.

```
**Algorithm** $A_\alpha(s)$:
*Step 0.* Let $j = 0$.
*Step 1.* If the next job $J_{j+1}$ is the *last* job, assign it to Machine 2 and stop. Otherwise, go to Step 2.
*Step 2.* If $M_2(j) + p_{j+1} > \alpha(s)(M_1(j) + sp_{j+1})$, assign $J_{j+1}$ to Machine 1; otherwise assign it to Machine 2. Let $j = j + 1$, go to Step 1.
```

For convenience of notation, we sometimes use β for $\beta(s)$ and α for $\alpha(s)$ throughout the paper. The following observation holds regardless of the exact definition of α.

Observation 1 *Let n be the number of jobs presented to the Algorithm A_α. Consider the resulting schedule produced for some $0 \leq j \leq n - 1$. Then $M_2(j) \leq \beta M_1(j)$ holds for this schedule.*

Proof. Since $M_1(0) = M_2(0) = 0$, the property holds in the beginning of the sequence. Therefore we can assume that $j \geq 1$. As long as machine 2 does not receive jobs, the property is true. Let J_t be the last job assigned to Machine 2 at a given moment (after the arrival of j jobs). From the algorithm, we have $M_2(t-1) + p_t \leq \alpha(M_1(t-1) + sp_t)$, since J_t was assigned to machine 2. Note that $p_t \leq M_2(t) = M_2(j)$, and the load of machine 1 is non-decreasing in time. Thus,

$$M_2(j) = M_2(t) = M_2(t-1) + p_t \leq \alpha(M_1(t-1) + sp_t) \leq \alpha(M_1(j) + sM_2(j)). \quad (1)$$

Since $s\alpha < 1$, so we have $M_2(j) \leq \frac{\alpha}{1 - s\alpha} M_1(j) = \beta M_1(j)$. ■

3 The Max-min objective

In this section our objective is to maximize the minimum completion time. Let $R(s) = \frac{3s+1+\sqrt{5s^2+2s+1}}{2s+2}$. We show that for nearly every value of s, $R(s)$ is the exact competitive ratio for the speed ratio s.

3.1 Analysis of the algorithm for the Max-min problem

In this section we use $\alpha(s) = \frac{\sqrt{5s^2+2s+1}-(s+1)}{2s^2}$. Note that α is the positive solution of $s^2\alpha^2 + (s+1)\alpha - 1 = 0$, which is equivalent to $\frac{1+s\alpha}{\alpha(s+1)} = \frac{1}{1-s\alpha}$. The above choice of α gives the value $R(s)$ to these two expressions. Using simple algebra, it is possible to show that $0 < \alpha < \frac{1}{s+1}$.

Theorem 2 *The competitive ratio of Algorithm $A_\alpha(s)$ is at most $R(s) = \frac{3s+1+\sqrt{5s^2+2s+1}}{2s+2}$.*

Proof. Let y be the processing time of the last job J_n. Then the cost of the algorithm C_{A_α} is C_{A_α} is $\min\{M_1(n) = M_1(n-1), M_2(n) = M_2(n-1)+y\}$. Consider the following two cases.

Case 1. $C_{A_\alpha} = M_2(n-1)+y$. Since $M_1(n) \geq M_2(n)$, machine 1 is non-empty. Let x be the processing time of the last job assigned to Machine 1, and t its index. From the algorithm, we have $M_2(n-1)+x \geq M_2(t-1)+x > \alpha M_1(t) = \alpha M_1(n-1)$ (otherwise this job would have been scheduled on Machine 2). Recall that J_n is a job with the longest processing time y. We have $y \geq x$ and thus we get $M_1(n-1) < \frac{1}{\alpha}(M_2(n-1)+y)$, therefore,

$$\begin{aligned} C^* &\leq \frac{M_1(n-1)/s + M_2(n-1)+y}{1+1/s} = \frac{M_1(n-1)+s(M_2(n-1)+y)}{s+1} \\ &\leq \frac{1/\alpha + s}{s+1}(M_2(n-1)+y) = \frac{3s+1+\sqrt{5s^2+2s+1}}{2s+2}C_{A_\alpha} \end{aligned}$$

Case 2. $C_{A_\alpha} = M_1(n-1)$. If $y \geq M_1(n-1)+sM_2(n-1)$, then the machines cannot be totally balanced. We apply Observation 1.

$$\begin{aligned} C^* = M_1(n-1)+sM_2(n-1) &\leq \left(1+s\frac{\alpha}{1-s\alpha}\right)M_1(n-1) \\ = \frac{1}{1-s\alpha}C_{A_\alpha} &= \frac{3s+1+\sqrt{5s^2+2s+1}}{2s+2}C_{A_\alpha}. \end{aligned}$$

Now assume that $y < M_1(n-1)+sM_2(n-1)$.

$$\begin{aligned} C^* &\leq \frac{M_1(n-1)/s + M_2(n-1)+y}{1+1/s} \\ &\leq \frac{M_1(n-1)/s + M_2(n-1) + M_1(n-1) + sM_2(n-1)}{1+1/s} \\ &= M_1(n-1)+sM_2(n-1) \leq \frac{1}{1-s\alpha}M_1(n-1) = \frac{3s+1+\sqrt{5s^2+2s+1}}{2s+2}C_{A_\alpha}. \end{aligned}$$

■

3.2 Lowers bounds

In this section, we give instances to show a lower bound which matches the upper bound we have given in the previous subsection for almost every value of s.

Lemma 3 *For any online algorithm A, the competitive ratio $R_A \geq \frac{3s+1+\sqrt{5s^2+2s+1}}{2s+2}$, when $s > 4+3\sqrt{2} = 8.24264$ or $1 \leq s \leq \frac{5}{3} + \left(\frac{367}{54} - \frac{5\sqrt{69}}{18}\right)^{\frac{1}{3}} + \left(\frac{367}{54} + \frac{5\sqrt{69}}{18}\right)^{\frac{1}{3}} \approx 5.40431$.*

Proof. To prove the lemma we consider three cases which are, $s = 1$, $1 < s < 2$ and $s \geq 2$. The first case can be easily combined into the second one, however, we give it separately to avoid confusion when referring to "the fast machine" and "the slow machine". Since the sequence must notify which job is last, we specify it where necessary. If not specified, it means the job is not last.

Case a. $s = 1$. We show a lower bound of $1 + \sqrt{2}/2$. We use the same sequence as in the proof in [17] for two identical machines, and the Min-max problem. The first two jobs have processing time 1.

sub-case a.1. They are assigned to different machines. Then the *last* job which has processing time 2 arrives. $C^* = 2$, $C_A = 1$. Thus $R_A \geq 2$.

sub-case a.2. They are assigned to the same machine, without loss of generality, assume it is M_1. The next job with processing time $x = \sqrt{2}$ arrives.

sub-case a.2.1. It is assigned to M_1, then the *last* job has processing time x. We know that $C^* = x + 1$ and $C_A = x$. Thus $R_A \geq \frac{x+1}{x} = 1 + \sqrt{2}/2$.

sub-case a.2.2. It is assigned to M_2, we add a *last* job of processing time $2 + x$. Clearly $C^* = 2 + x$ and $C_A \leq 2$. Which implies that $R_A \geq \frac{2+x}{2} = 1 + \sqrt{2}/2$.

Case b. $1 < s \leq 2$. We start with a job of size $p_0 = 1$. We use a similar pattern for both cases (each case relates to the machine where the first job is assigned). We use a parameter $0 < z(s) < 1$ (the value of z depends on the machine which received the first job, and is fixed later). For $k = 1, \cdots, K+1$, we issue the job $p_k = z(1+z)^{k-1}$. Clearly, the sum of all jobs up to job j, is $(1+z)^j$. After this sequence we give another *last* job of size $p_{K+2} = p_{K+1}$. The total sum of jobs is $S_K = (2z+1)(z+1)^K$. Note that the last job is indeed the largest since starting the third job, the sequence is non-decreasing. We would like to show that if K is large enough, then the optimal schedule is almost balanced. We start with the following claim.

Claim 4 *Given the set $U = \{1, z, z(1+z), \cdots, z(1+z)^j\} = \{p_0, \cdots, p_{j+1}\}$. Given a number y such that $y \in [0, (1+z)^{j+1}]$ (i.e. y is positive and smaller or equal than the sum of all elements in U), then we can choose a subset of W, $W \subseteq U$ such that $|y - \sum\limits_{i \in W} p_i| \leq 1$.*

Proof. By induction. If $j = 0$, then $y \in [0, 1+z]$ and the possible sums are $\{0, z, 1, 1+z\}$. The largest gap between two consecutive elements is $\max\{z, 1-z\} \leq 1$, therefore it is possible to choose the required subset. Given $j > 0$, we choose the subset in the following way. If $y \geq z(1+z)^j$, we choose $z(1+z)^j$ and set $y' = y - z(1+z)^j$. This gives $0 \leq y' \leq (1+z)^j$. Otherwise we do not choose $z(1+z)^j$ and set

$y' = y$. In both cases we can continue using the inductive hypothesis for $j-1$. Note that in the second case $y = y' < z(1+z)^j \leq (1+z)^j$ since $z \leq 1$. ■

Next we show how to use this claim to get a convenient offline schedule.

Claim 5 *Given some $\varepsilon > 0$, it is always possible to choose K such that we can always assign the jobs $p_1, \ldots, p_{K+2}$ such that $\frac{S_K}{1+\frac{1}{s}} \leq C^*(1+\varepsilon)$ is satisfied. I.e., the schedule is almost flat.*

Proof. In a flat schedule, the loads both machines would be $\frac{S_K}{1+\frac{1}{s}}$. We always assign the last job to the fast machine. The amount that the fast machine still needs to achieve a flat schedule is

$$\begin{aligned} \frac{S_K}{(1+\frac{1}{s})} - p_{K+2} &= \frac{s(2z+1)(z+1)^K}{s+1} - z(z+1)^K \\ &= (z+1)^K \left(\frac{s(2z+1)}{s+1} - z \right) \\ &= \frac{(z+1)^K (zs+s-z)}{s+1}. \end{aligned}$$

This value is positive since $z \leq 1 < s$. It is easy to show $\frac{S_K}{1+\frac{1}{s}} - p_{K+2} \leq (z+1)^{K+1}$, thus the difference satisfies that conditions of the previous claim. therefore there exists a subset of jobs we can add to the fast machine getting load of at least $\frac{S_K}{1+\frac{1}{s}} - 1$ and at most $\frac{S_K}{1+\frac{1}{s}} + 1$. This means that the workload of the slow machine which receives all other jobs is at least $\frac{S_K}{1+\frac{1}{s}} - s$ and at most $\frac{S_K}{1+\frac{1}{s}} + s$. We get $C^* \geq \frac{S_K}{1+\frac{1}{s}} - s$. Since S_K can be arbitrarily large, we choose K such that $S_K \geq (s+1)(1+\frac{1}{\varepsilon})$. ■

We can proceed with the proof now. If the first job is assigned to the fast machine we use $z(s) = R(s)/s - 1$, where $R(s) = \frac{3s+1+\sqrt{5s^2+2s+1}}{2s+2}$, which is a solution of the equation $(s+1)R^2 - (3s+1)R + s = 0$. It is simple to show $s < R(s) < 2 \leq s+1$ for the case $s < 2$. We get that $sz(s) = R(s) - s$, so $0 < z(s) \leq sz(s) < 1$.

Choose ε and let K be an integer satisfying claim 5. We give the jobs. If some job p_{j+1} is assigned to the slow machine, as a result, this machine has the workload $sz(1+z)^j$, and the workload of the fast machine is $(1+z)^j$. Thus we stop the sequence and give a last job of size $s(1+z)^{j+1}$ (Clearly this is the largest job, since it is larger than the sum of all previous jobs). Since $C^* = s(1+z)^{j+1}$, and $C_A = (1+z)^j$ (using $sz < 1$), which gives the competitive ratio $s(1+z) = R$.

Upon arrival of job $K+2$ (which is last), we have $C^* \geq \frac{S_K}{(1+\frac{1}{s})(1+\varepsilon)}$. We get a workload of $sz(1+z)^K < (1+z)^K$ on the slow machine, and a workload of $(z+1)^{K+1}$ on the fast one. So, the ratio $R_A = \frac{C^*}{C_A} \geq \frac{2z+1}{z(s+1)}/(1+\varepsilon)$.

We get $R_A = \frac{2R/s-1}{(s+1)(R/s-1)} = \frac{2R-s}{(s+1)(R-s)}$. In the following, we will show that $R_A \geq R$. This is equivalent to showing $2R - s \geq (s+1)(R-s)R = R^2(s+1) - s(s+1)R$. From the definition of R, we get $2R - s \geq (3s+1)R - s - s(s+1)R$, this gives $2R - s \geq R(-s^2+2s+1) - s$ or $R(s^2-2s+1) \geq 0$ which clearly holds.

If the first job is assigned to the slow machine we use $z(s) = R(s) - 1$, Since $s < R(s) < 2$ for the case $s < 2$. We get that $0 < z(s) < 1 < s$.

Choose ε and let K be an integer satisfying claim 5. We give the jobs. If some job p_{j+1} is assigned to the fast machine it has the load $z(1+z)^j$, and the load of the slow machine is $s(1+z)^j$. Thus we stop the sequence and give a last job of size $s(1+z)^{j+1}$. Since $C^* = s(1+z)^{j+1}$, and $C_A = s(1+z)^j$ (using $z < s$), which give the competitive ratio $1 + z(s) = R(s)$.

After the arrival of job $K+2$ we have $C^* \geq S_K/(1+\frac{1}{s})/(1+\varepsilon)$. We get a load of $z(1+z)^K$ on the fast machine, and $s(z+1)^{K+1}$ on the slow one. So, the ratio $R_A = \frac{C^*}{C_A} \geq \frac{s(2z+1)}{z(s+1)}/(1+\varepsilon)$.

We get $R_A = \frac{s(2R-1)}{(s+1)(R-1)} = R(s)$ due to the definition of $R(s)$ as the solution of $(s+1)R^2 - (3s+1)R + s = 0$.

Case c. $2 < s \leq s_1 = \frac{5}{3} + \left(\frac{367}{54} - \frac{5\sqrt{69}}{18}\right)^{\frac{1}{3}} + \left(\frac{367}{54} + \frac{5\sqrt{69}}{18}\right)^{\frac{1}{3}} \approx 5.40431$ and $s > 4 + 3\sqrt{2} \approx 8.2426$. We consider the following instance: given the sequence of four jobs with size of $\{1, z, x(1+z), x(1+z)\}$, respectively. Where $x = R(s) - 1$, again let $R(s) = \frac{3s+1+\sqrt{5s^2+2s+1}}{2s+2}$. It is interesting to note that $R(s)$ is a non-decreasing function of s. $2 \leq R(s) \leq (3+\sqrt{5})/2$ in the case that $s \geq 2$. For simplicity, we use R instead of $R(s)$ without cause any confuse. The value of z depends on s,

$$z = \begin{cases} \frac{2x+1}{s-2x}, & if \;\; 2 < s \leq s_1; \\ \frac{s+1}{2R-1} - 1, & if \;\; s > 4 + 3\sqrt{2}. \end{cases}$$

In the following we will prove that $z \geq x$ in this case. First, we consider $2 < s \leq s_1$. $z = \frac{2x+1}{s-2x} = \frac{2R-1}{s+2-2R}$. $s + 2 - 2R > 0$, since $s^3 \geq 3s + 2$ if $s \geq 2$. In order to show $z \geq R - 1 = x$, we only need to show that $2R - 1 \geq s(R-1) - 2(R-1)^2$, which is equivalent to $s \leq 2(R-1) + \frac{1}{R-1} + 2$. This holds for $s \leq s_1$.

Now we consider $s > 4 + 3\sqrt{2}$. From the definition of R, we have $z = \frac{s+1}{2R-1} - 1 = \frac{s}{R(R-1)} - 1$. In order to show $z \geq x = R - 1$, we only need to prove that $s \geq 2R^2 - R - 1$. This holds $s \geq 4 + 3\sqrt{2}$. We can proceed the proof of this case now. We start with job J_1 of size 1.

Case c.1. J_1 is assigned to M_2. Then the next job is claimed to be the last job of size s. We have $C^* = s$ and $C_A = 1$, thus $R_A \geq \frac{C^*}{C_A} = s > R$, when $s > 2$.

Case c.2. J_1 is assigned to M_1. The next job J_2 of size z arrives.

Case c.2.1 J_2 is assigned to M_2. Then the next job of size $s(1+z)$ is claimed to be the last job. We have $C^* = s(1+z)$, $C_A = \max\{s, z\}$. Thus $R_A \geq \max\{1+z, s+s/z\} \geq R$, from $z \geq x = R-1$.

Case c.2.2 J_2 is assigned to M_1. Then the next job J_3 of size $x(1+z)$ arrives.

Case c.2.2.1 J_3 is assigned to M_2. Then the last job of size $s(1+z) + sx(1+z)$ arrives. Clearly, $C^* = s(1+z) + sx(1+z)$ and $C_A = s(1+z)$, since $s \geq R > x$. Thus $R_A \geq 1 + x = R$.

Case c.2.2.2 J_3 is assigned to M_1. The last job of size $x(1+z)$ arrives. Clearly, $C_A = x(1+z)$. Now we consider the optimal value C^*. In the case $2 < s \leq 2+2\sqrt{2}$, assign job J_2 of size z to slow machine, and all the other jobs to fast machine. Then $C^* = \min\{sz, 1 + 2x(1+z)\}$, from the definition of $z = \frac{2x+1}{s-2x}$, we have $sz = 1 + sx(1+z)$. Then $R_A = \frac{sz}{x(1+z)} = \frac{s(2x+1)}{x(s+1)}$. Note that $s(2x+1) = (s-1)x + s + x(s+1)$. From the definition of x, $(s+1)x^2 - (s-1)x - s = 0$, $s(2x+1) = (s+1)x^2 + x(s+1) = (1+x)x(s+1)$.

Then $R_A = \frac{s(2x+1)}{x(s+1)} = 1 + x = R$. Now we consider the optimal value C^* in the case $s > (9 + 5\sqrt{5})/2$. Just assign the first job of size 1 to slow machine and all the other jobs to the fast machine. Therefore, $C^* = \min\{s, 2x(1+z)+z\}$. From the definition of z, we have $s = 2x(1+z)+z$. Then, $R_A = \frac{s}{x(1+z)} = \frac{s}{(R-1)(1+s/(R(R-1))-1)} = R$. ■

Lemma 6 *For any online algorithm A, the competitive ratio $R_A \geq \max\{(\frac{1}{2} + \frac{\sqrt{69}}{18})^{\frac{1}{3}} + (\frac{1}{2} - \frac{\sqrt{69}}{18})^{\frac{1}{3}} + 1 \approx 2.3247, R_1(s) = (\frac{1}{27} + \frac{s}{2} + \frac{\sqrt{12s+81s^2}}{18})^{\frac{1}{3}} + (\frac{1}{27} + \frac{s}{2} - \frac{\sqrt{12s+81s^2}}{18})^{\frac{1}{3}} + \frac{1}{3}\}$ for any $\frac{5}{3} + \left(\frac{367}{54} - \frac{5\sqrt{69}}{18}\right)^{\frac{1}{3}} + \left(\frac{367}{54} + \frac{5\sqrt{69}}{18}\right)^{\frac{1}{3}} \approx 5.40431 = s_1 \leq s \leq 4 + 3\sqrt{2}$.*

Proof. Let $2 \leq \hat{R} \leq R(s)$ be a competitive ratio we would like to prove. We start with a job of size 1. If it is assigned to M_2, then a last job of size s arrives. $C^* = s$ and $R_A \geq s \geq R(s)$. Assume therefore that the first job is assigned to M_1, the next job is of size $\hat{R} - 1$. If the second job is assigned to M_2, then a last job of size $s\hat{R}$ appears. Thus $C^* = s\hat{R}$ and $C_A = \max\{\hat{R} - 1, s\} = s$, $R_A \geq \hat{R}$. If the second job is assigned to M_1, then the next job is of size $\hat{R}(\hat{R} - 1) \geq \hat{R}$. If it is assigned to M_2, then a last job of size $s\hat{R}^2$ arrives. We have $C^* = s\hat{R}^2$ and $C_A = s\hat{R}$, thus again $R_A \geq \hat{R}$. Finally, if all three jobs are assigned to M_1, a last job of size $\hat{R}^2 - \hat{R}$ arrives.

At this time, we consider two possible offline schedules.

In the first option, all jobs but the first one of size $\hat{R} - 1$ are assigned to the fast machine. In this case $C^* = 2\hat{R}^2 - 2\hat{R} + 1$. To show that, note that the other machine has the workload $s(\hat{R} - 1)$. Comparing the values we have that $s(\hat{R} - 1) \geq 2\hat{R}^2 - 2\hat{R} + 1$ for $s \geq s_1$. Therefore the competitive ratio is at least $\frac{2\hat{R}^2 - 2\hat{R} + 1}{\hat{R}^2 - \hat{R}}$. Taking $\hat{R}$ to be the solution of the equation $x^3 - 3x^2 + 2x - 1 = 0$, we achieve the first option for the lower bound.

In the second option, all jobs but the first one of size 1 are assigned to the fast machine. In this case $C^* = s$. To show that, note that the sum of all other jobs is $2\hat{R}^2 - \hat{R}$. Since $\hat{R} \geq 2.32$, we have $2\hat{R}^2 - \hat{R} \geq 8.44$ which is strictly larger than s. We have $C_A = \hat{R}^2 - \hat{R}$, and therefore the competitive ratio is at least $\frac{s}{\hat{R}^2 - \hat{R}}$. Taking $\hat{R}$ to be the solution of the equation $x^3 - x^2 = s$, we achieve the second option for the lower bound.

The breakpoint between the two lower bounds is $R^3(s_1) - R^2(s_1) \approx 7.159191247$. The first lower bound is larger in the first interval, and the second one, in the second interval. ■

For $s > 4 + 3\sqrt{2}$ and $1 \leq s \leq s_1$, the algorithm $A_\alpha(s)$ is a best possible online algorithm. As can be seen in Figure 1, the gap between the lower and upper bounds in the remaining interval is relatively small.

4 The Min-max objective

In this section we deal with this semi-online problem on two uniform machines whose objective is to minimize the makespan.

4.1 Lower Bounds

In this section we give lower bounds depend on the speed s.

Lemma 7 *For any semi-online algorithm A, no deterministic algorithm can have competitive ratio lower than*

$$R_A = \begin{cases} 1+\frac{1}{s}, & if \;\; s \geq 1+\sqrt{3}; \\ 1+\frac{2s}{2+2s+s^2}, & if \;\; 2 \leq s < 1+\sqrt{3}; \\ 1+\frac{s}{1+s^2}, & if \;\; 1.46557 < s < 2; \\ R'(s) = \frac{s-1+\sqrt{5s^2+2s+1}}{2s}, & if \;\; 1 < s \leq 1.46557. \end{cases}$$

Proof. We prove the lemma by the following cases.

Case a. $s \geq 1+\sqrt{3}$. We start with a job J_1 of processing time 1. If J_1 is scheduled on the fast machine M_2, then the last job, which has size s arrives. Clearly, $C_A = \min\{s^2, 1+s\} = 1+s$ and $C^* = s$. If J_1 is scheduled on the slow machine M_1, the last job is of size 1. We have $C^* = 2$, however, $C_A = s$. Since $s \geq 1+\sqrt{3}$, we have $s^2-2s-2 \geq 0$, this is equivalent to $1+\frac{1}{s} \leq s/2$. Therefore, $R_A \geq 1+\frac{1}{s}$ in this case.

Case b. $1+\sqrt{3} > s \geq 2$.

Let $x = \frac{2s-2s^2-s^3}{-4-6s+s^3}$ and $y = s(1+x)/2$. We have $x \geq 1$ when $1.8832 \leq s < 1+\sqrt{3}$, $x \geq s/2$ when $s \geq 1.7511$ and $y \geq x$ when $0 < s < 1+\sqrt{3}$.

Again, we start with first job J_1 of size 1.

Case b.1. J_1 is assigned to fast machine M_2, then the last job of size s arrives, which implies that $R_A \geq 1+\frac{1}{s}$.

Case b.2. J_1 is assigned to M_1, then next job J_2 of size x arrives.

Case b.2.1. J_2 is assigned to M_1, the last job J_3 of size x arrives. We have $C^* = 2x$ since $s \leq 2x$ for $s \geq 1.7511$ and $C_A = s(1+x)$. So $R_A = s(1+x)/(2x) = 1+4/(s^2+2s-2) \geq 1+2s/(2+2s+s^2)$ in our interval.

Case b.2.2. J_2 is assigned to M_2, next job J_3 of size y arrives.

Case b.2.2.1. J_3 is assigned to M_1, the last job is of size y. Then $C_A = s(1+y)$, $C^* = 2y$. Then $R_A = 1+2s/(2+2s+s^2)$.

Case b.2.2.2. J_3 is assigned to M_2, the last job J_4 of size $s(1+x+y)$ arrives. We have $C^* = s(1+x+y)$ and $C_A = x+y+s(1+x+y)$. Hence, $R_A = 1+(x+y)/(s(1+x+y)) = 1+2s/(2+2s+s^2)$.

Case c. $\beta \approx 1.46557 < s < 2$. Where $\beta \approx 1.46557$ is a solution of the equation of $s^3-s^2-1=0$. The lower bound we prove here is smaller than $\min\{s, 1+\frac{1}{s}\}$ for all considered values of s.

We start with a sub-interval which is $s \leq 1.8832 = \frac{1}{2}+\frac{\sqrt{2}}{2}+\frac{1}{2}\sqrt{-1+2\sqrt{2}}$. If the first job is assigned to the fast machine, then the last job has size of s. Thus, in this case we have always $R_A \geq \min\{s, 1+\frac{1}{s}\}$. Consider now the case that the first job is assigned at slow machine M_1.

Let $z = s + 1/s$. The next job is of size z. If it is assigned to M_1, the last job is of size z as well. $C^* = sz$ for $sz \le 1 + z$ and $1 + z$ otherwise. $C_A \ge s(1 + z)$. This gives a lower bound of at least $1 + \frac{s}{s^2+1}$.

Otherwise, the next job is of size $x = (2 + 2s^2 - s^3)/(s^3 - s^2 - 1)$, this number is non-negative in our interval. If it is assigned to the fast machine, the last job is of size $s(1+x+z)$. We have $C^* = s(1+x+z)$. If it is assigned to the slow machine, this results in a competitive ratio larger than s. Otherwise the ratio is $1 + \frac{x+z}{s(x+z+1)} \ge 1 + \frac{s}{s^2+1}$ for $s \le 1.8832$.

If x is assigned to the slow machine, the next job is $y = (1+s^2)^2/(s(s^3 - s^2 - 1))$. Note that $y \ge x$ and $y \ge z$ in the interval $s \le 2$. If y is assigned to the fast machine, we have the last job $s(x + y + 1 + z) = C^*$. If it is assigned to the slow machine, this results in a competitive ratio larger than s. Otherwise the ratio is $1 + \frac{y+z}{s(x+y+z+1)} = 1 + \frac{s}{s^2+1}$. Otherwise, the last job is of size y as well. We get that $C^* = sy = y + x + 1 + z$. $C_A \ge s(1 + x + y)$.

We would like to use a similar instance for the rest of the interval. However, the proof fails at the point where the job of size x arrives. Instead of the above sequence, we let the first job be of size $1 + x$ instead of 1. Clearly, if it is assigned to the fast machine we still get a high competitive ratio. Otherwise, the next job is of size z. If we reach a situation where this job is assigned to the fast machine, we can continue as before, since all optimal schedules assigned the jobs of sizes 1 and x to the same machine. It is only left to consider the case where the job of size z is assigned to the slow machine, we let another job of size z arrive and it is the last job. Note that $z \ge 1 + x$ already for $x \ge 1.8393$. Since $sz > z + x + 1$ in all the interval we are considering now, we have $C^* \le sz$. This gives a ratio $\frac{x+z+1}{z} = 1 + \frac{s}{s^3-s^2-1} > 1 + \frac{s}{s^2+1}$ for $s \le 2.3593$.

Case d. $1 < s \le \beta \approx 1.46577$. Where β is a solution of the equation $s^3 = s^2 + 1$. Before proving this case, we first prove the following claim regarding the first job.

Claim 8 *The first job must be assigned to the slow machine M_1, otherwise we can get an instance I for which the ratio is no less than $R'(s)$.*

Proof. Suppose that the first job is assigned to the fast machine M_2. The instance I is given as follows. Assume that this first job has size 1 (otherwise scale accordingly). We continue similarly to Lemma 3 with jobs of sizes $z, z(1 + z), z(1 + z)^2, \ldots$ where $z = (s + 1 - s^2)/(s^2) < 1$. The sum after i such jobs is $(1 + z)^i$. If some job in this sequence is assigned to the slow machine, we can immediately give a last job of size $s(1 + z)^i$ (instead of the remainder of this sequence), and thus have $C^* = s(1 + z)^i$. If this additional job is assigned to the slow machine, it contains the last two jobs and we get a ratio of $(sz(1+z)^{(i-1)} + s^2(1+z)^i)/(s(1+z)^i) = (z+s(z+1))/(1+z) = (2s+1)/(s+1) > R'(s)$. Otherwise, all jobs but one (the second to last job) are on the fast machine and we get the ratio $(s(1 + z)^i + (1 + z)^{(i-1)})/(s(1 + z)^i) = (s + sz + 1)/(s + sz) = (2s + 1)/(s + 1)$.

After giving enough jobs (that are all assigned to the fast machine), we can balance any required schedule. Denote the sum of jobs presented so far by X. Next we have a job of size TX where $T = (s + 1 - sR'(s))/(2sR'(s) - s - 1)$. The largest job so far (not including the new job) has size $Xz/(1 + z)$ so in order to let the last job have the same size as this job, the new job need to be the largest, and we need to have $T \geq z/(1+z)$. It can be easily verified that this algebraic property holds in the interval considered here. If this job of size TX is assigned to the fast machine, a last job of size TX follows it. We get $C^* = (2TX + X)/(1 + 1/s)$ and the fast machine has total size of jobs which is at least $X + TX$. Thus we get the ratio $(s+1)(T+1)/(s(2T+1)) = R'(s)$

If the job of size TX was assigned to the slow machine, a last job of size $s(T+1)X$ follows it. We have $C^* = s(T+1)X$. This job may be assigned to the fast machine or the slow machine. In the first case, this machine additionally has all smaller jobs and we get the ratio $(X+s(T+1)X)/(s(T+1)X) = 1+1/(s(T+1))$. In the second case, we get the ratio $(s^2(T+1)X + sTX)/(s(T+1)X) = (T + s(T+1))/(T+1)$. The last two ratios are larger than $R'(s)$. ■

Now we are ready to prove this case. We abuse the notation z and define the value $z(s) = \frac{s(R'(s)-1)}{1+s-sR'(s)}$. Note that $1 + s > sR'(s)$ for any $s < 2$. We further define $\alpha = \frac{z+1-sz}{s+1}$. The value of α is monotonically decreasing with s and we have $0 \leq \alpha \leq \frac{1}{2}$ in this interval.

The first job is of size $\alpha \geq 0$. If $\alpha = 0$ the sequence starts with the next job. Otherwise, it must be assigned to M_1 according to Claim 8.

The next job is of size $1 - \alpha$. This job must be assigned to M_1 otherwise a last job of size s arrives, the $C^* = s$ and $C_A = \min\{1 - \alpha + s, \alpha s + s^2\}$. Since $\alpha + s = R'(s)$ and $1 - \alpha + s \geq sR'(s)$ hold in our interval.

The next job is of size z. This job must be assigned to M_1 otherwise a last job of size $s(z+1)$ arrives, we have $C^* = s(z+1)$, $C_A = \min\{z + s(z+1), s + s^2(z+1)\}$. Again, we get these two ratios is at least $R'(s)$ in our interval.

The last job is of size z as well. According to the value of α, we have that $C^* = z + 1 - \alpha = s(\alpha + z)$. However $C_A \geq s(1+z)$, and the ratio $\frac{s(1+z)}{s(\alpha+z)} = R'(s)$. ■

As can be seen in Figure 2, the gap between the lower and upper bounds in the remaining interval is also relatively small.

4.2 Analysis of the algorithm for the Min-max problem

In this section, we present algorithms for the above semi-online problem. The competitive ratios again depend on the value of s. First we consider a naïve algorithm which just uses the fast machine for all jobs. In [7], it was shown that the competitive ratio of this algorithm is at most of $1 + 1/s$ (this is a simple fact that follows from total size of all jobs).

For $s < 2$, we schedule the jobs according to the following algorithms. Let $\alpha(s) = \frac{\sqrt{5s^2+2s+1}-(s+1)}{2s}$. This is the positive solution of the equation $s\alpha^2 + (s+1)\alpha - s = 0$. With this value of α, we get that the two expressions $\frac{s+1}{1+s\alpha}$ and $1 + \alpha$ are equal, and have the value $R'(s)$.

Then $s\alpha(s) < 1$ when $s < 2$. Let $\beta(s) = \frac{\alpha(s)}{1-s\alpha(s)}$. We use $A_\alpha(s)$ for our problem, and prove the following theorem.

Theorem 9 *The competitive ratio of Algorithm A_α is at most $R'(s) = \frac{s-1+\sqrt{5s^2+2s+1}}{2s} = 1 + \alpha$.*

Proof. Let y be the processing time of the last job J_n. Then the cost of the algorithm C_{A_α} is $\max\{M_1(n) = M_1(n-1), M_2(n) = M_2(n-1) + y\}$. Consider the following two cases.

Case 1. $C_{A_\alpha} = M_1(n-1)$. Let x be the processing time of the last job assigned to Machine 1, and t its index. From the algorithm, we have $M_2(n-1) + x \geq M_2(t-1) + x > \alpha M_1(t) = \alpha M_1(n-1)$ (otherwise this job would have been scheduled on Machine 2). Since J_n is a job with the longest processing time, $y \geq x$. We get $\alpha(s) M_1(n-1) < M_2(n-1) + y$. Therefore,

$$\begin{aligned} C^* &\geq \frac{M_1(n-1)/s + M_2(n-1) + y}{1 + \frac{1}{s}} = \frac{M_1(n-1) + s(M_2(n-1) + y)}{s+1} \\ &\geq \frac{1 + s\alpha}{s+1}(M_1(n-1)) \end{aligned}$$

Then $R_{A_\alpha} = \frac{C_{A_\alpha(s)}}{C^*} \leq \frac{s+1}{1+s\alpha(s)} = \frac{s-1+\sqrt{5s^2+2s+1}}{2s}$.

Case 2. $C_{A_\alpha} = M_2(n-1) + y$. If $y \geq M_1(n-1) + sM_2(n-1)$, then $C^* \geq y$. Using Observation 1 we have $M_1(n-1) \geq \frac{1}{\beta} M_2(n-1)$, thus,

$$\begin{aligned} R_{A_\alpha} &\leq \frac{M_2(n-1) + y}{y} = 1 + \frac{M_2(n-1)}{y} \\ &\leq 1 + \frac{M_2(n-1)}{M_1(n-1) + sM_2(n-1)} \\ &= 1 + \frac{M_2(n-1)}{sM_2(n-1) + \frac{1-s\alpha}{\alpha} M_2(n-1)} \\ &= 1 + \alpha. \end{aligned}$$

Next we consider the case that $y < M_1(n-1) + sM_2(n-1)$. Then $C^* \geq \frac{M_2(n-1)+y+\frac{M_1(n-1)}{s}}{1+\frac{1}{s}} = \frac{M_1(n-1)+sy+sM_2(n-1)}{s+1}$. We use Observation 1 once again,

$$\begin{aligned} R_{A_\alpha} &\leq \frac{(s+1)(M_2(n-1) + y)}{sM_2(n-1) + sy + M_1(n-1)} = 1 + \frac{1}{s} - \frac{(s+1)M_1(n-1)}{sM_1(n-1) + s^2M_2(n-1) + s^2y} \\ &\leq 1 + \frac{1}{s} - \frac{(s+1)M_1(n-1)}{(s^2+s)M_1(n-1) + (s^3+s^2)M_2(n-1)} \\ &= 1 + \frac{1}{s} - \frac{M_1(n-1)}{sM_1(n-1) + s^2M_2(n-1)} \leq 1 + \frac{1}{s} - \frac{1}{s + s^2\beta} = 1 + \alpha. \end{aligned}$$

■

It is easy to see that $R(s) < 1 + \frac{1}{s}$ when $s < 2$. Thus the algorithm A_α has a better performance for $s < 2$. Note that for $s \geq 1 + \sqrt{3}$, we have presented a best possible algorithm. Algorithm A_α is also optimal for identical machines, i.e. when $s = 1$, in this case our algorithm reduces to the one of [17].

Acknowledgements. We would like to thank an anonymous referee for many helpful suggestions to improve the presentation of this paper.

References

[1] E. Angelelli, A. Nagy, M. Speranza and Z. Tuza. The on-line multiprocessor scheduling problem with known sum of the tasks. *Journal of Scheduling* **7** (2004), 421-428.

[2] Y. Azar and O. Regev. Online bin stretching. *Theorectial Computer Science* **268** (2001), 17-41.

[3] G. Dósa and Y. He. Semi-Online algorithms for parallel machine scheduling problems. *Computing* **72** (2004), 355-363.

[4] L. Epstein. Bin Stretching revisted. *Acta Informatica* **39(2)** (2003), 97-117.

[5] L. Epstein. Tight bounds for bandwidth allocation on two links. *Discrete Applied Math.* **148(2)** (2005), 181-188.

[6] L. Epstein and L. M. Favrholdt. Optimal non-preemptive semi-online scheduling on two related machines. *Journal of Algorithms*, 57(1): 49–73, 2005.

[7] L. Epstein, J. Noga, S.S. Seiden, J. Sgall and G.J. Woeginger. Randomized Online Scheduling on Two Uniform Machines. *Journal of Scheduling* **4(2)** (2001), 71-92.

[8] R. Graham. Bounds for certain multiprocessing anomalies. *Bell System Technical Journal* **45** (1966), 1563-1581.

[9] R. Graham. Bounds on multiprocessing timing anomalies. *SIAM Journal on Applied Mathematics* **17(2)** (1969), 416-429.

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

[11] 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.

[12] P. Sanders, N. Sivadasan, and M. Skutella. Online Scheduling with Bounded Migration. *Proceedings of Automata, Languages and Programming: 31st International Colloquium*, (ICALP 2004), Lecture Notes in Computer Science **3142** (2004),1111-1122.

[13] S. Seiden, J. Sgall, and G. Woeginger. Semi-online scheduling with decreasing job sizes. *Opereations Research Letters* **27** (2000), 215-221.

[14] J. Sgall. On-line scheduling. *Online algorithms – The State of Art*, Lecture Notes in Computer Science **1442** (1998), 196-231.

[15] Z. Tan and Y. He. Semi-on-line problems on two identical machines with combined partial information. *Operations Research Letters* **30** (2002), 408-414.

[16] G. Zhang. A simple semi on-line algorithm for $P2//C_{\max}$ with a buffer. *Information Processing Letters* **61** (1997), 145-148.

[17] G. Zhang and D. Ye. A note on on-line scheduling with partial information. *Computers & Mathematics with Applications* **44** (2002), 539-543.