ELSEVIER

Theoretical Computer Science 268 (2001) 133–143

Theoretical Computer Science

www.elsevier.com/locate/tcs

An optimal online algorithm for scheduling two machines with release times

John Noga[a,1], Steven S. Seiden[b,*,2]

[a] *Technische Universität Graz, Institut für Mathematik B, Steyrergaße 30/2. Stock, A-8010 Graz, Austria*

[b] *Louisiana State University, Department of Computer Science, 298 Coates Hall, Baton Rouge, LA 70803, USA*

Accepted May 2000

Abstract

We present a deterministic online algorithm for scheduling two parallel machines when jobs arrive over time and show that it is $\frac{1}{2}(5-\sqrt{5}) \approx 1.38198$-competitive. The best previously known algorithm is $(3/2)$-competitive. Our upper bound matches a previously known lower bound, and thus our algorithm has the best possible competitive ratio. We also present a lower bound of 1.21207 on the competitive ratio of any randomized online algorithm for any number of machines. This improves a previous result of $4-2\sqrt{2} \approx 1.17157$.

Keywords: Scheduling; Competitive online algorithms; Lower bounds

1. Introduction

We consider one of the most basic scheduling problems: scheduling parallel machines when jobs arrive over time with the objective of minimizing the makespan. This problem is formulated as follows: There are m machines and n jobs. Each job j has a *release time* r_j and a *processing time* p_j. An algorithm must assign each job to a machine and fix a *start time* s_j. No machine can run more that one job at a time and no job may start prior to being released. Jobs may not be preempted. We define the *completion time* for job j to be $C_j = s_j + p_j$. The *makespan* is $\max_j C_j$. The

* Corresponding author.

E-mail addresses: jnoga@opt.math.tu-graz.ac.at (J. Noga), sseiden@acm.org (S.S. Seiden).

[1] This research has been supported by the START program Y43-MAT of the Austrian Ministry of Science.

[2] This research was done while the author was at the Technische Universität Graz.

PII: S0304-3975(00)00264-4

algorithm's goal is to minimize the makespan. In the scheduling problem notation of Lawler et al. [14] this problem is $P|r_j|\max C_j$.

We study an *online* version of this problem, where jobs are completely unknown until their release times. In contrast, in the *offline* version all jobs are known in advance. Since it is in general impossible to solve the problem optimally online, we consider algorithms which approximate the best possible solution.

Competitive analysis is a type of worst case analysis where the performance of an online algorithm is compared to that of the optimal offline algorithm. This approach of analyzing online problems was initiated by Sleator and Tarjan, who used it to analyze the List Update problem [19]. The term competitive analysis originated in [13]. For a given job set σ, let $\mathrm{cost}_A(\sigma)$ be the cost incurred by an algorithm $\mathscr{A}$ on σ. Let $\mathrm{cost}(\sigma)$ be the cost of the optimal offline schedule for σ. A scheduling algorithm $\mathscr{A}$ is ρ-*competitive* if

$$\mathrm{cost}_{\mathscr{A}}(\sigma) \leqslant \rho\,\mathrm{cost}(\sigma),$$

for all job sets σ. The *competitive ratio* of $\mathscr{A}$ is the infimum of the set of values ρ for which $\mathscr{A}$ is ρ-competitive. Our goal is to find the online algorithm with smallest possible competitive ratio, the *optimal online* algorithm.

A great deal of work has been done on parallel machine scheduling when jobs are presented in a list and must be immediately scheduled prior to any processing [1–3, 5, 6, 8–10, 12, 16, 17]. On the other hand, very little work has been done on the version which we address here, despite the fact that this version of the problem seems more realistic. The known results are as follows: Hall and Shmoys show that the LIST algorithm of Graham is 2-competitive for all m [10, 11]. In [18], Shmoys et al. present a general online algorithm for scheduling with release dates. This algorithm uses as a subroutine an offline algorithm for the given problem. If the offline algorithm is an ρ-approximation, then the online algorithm is 2ρ-competitive. They also show a lower bound of $10/9$ on the competitive ratio for online scheduling of parallel machines with release times. The LPT algorithm starts the job with the largest processing time whenever a machine becomes available. Chen and Vestjens show that LPT is $3/2$ competitive for all m [7]. These same authors also show a lower bound of $3-\varphi \approx 1.38197$ for $m=2$ and a general lower bound of 1.34729, where $\varphi=(1/2)(1+\sqrt{5})\approx 1.61803$ is the golden ratio. Stougie and Vestjens [20] show a randomized lower bound of $4-2\sqrt{2}\approx 1.17157$ which is valid for all m.

We present an algorithm for scheduling two parallel machines, called SLEEPY, and show that it is $(3-\varphi)$-competitive, and thus has the best possible competitive ratio. We also show a randomized lower bound of

$$1+\max_{0\leqslant u<1}\frac{u}{1-2\ln(1-u)} > 1.21207.$$

Finally, we show that any distribution over two deterministic algorithms is at best $(5/4)$-competitive. It is our hope that these results arouse interest in the general m-machine problem.

2. The SLEEPY algorithm

We say that a job is *available* if it has been released, but not yet scheduled. A machine that is not processing a job is called *idle*.

Our algorithm, which we call SLEEPY, is quite simple. If both machines are processing jobs or there are no available jobs, then we have no choice but to wait. So, assume that there is at least one idle machine and at least one available job. If both machines are idle then start the available job with largest processing time. If at time t, one machine is idle and the other machine is running job j then start the available job with largest processing time if and only if $t \geqslant s_j + \alpha p_j$.

With this algorithm a machine can be idle for two reasons. If a machine is idle because $t < s_j + \alpha p_j$ we say the machine is *sleeping*. If a machine is idle because there are no available jobs we say that the machine is *waiting*. If a machine is waiting we call the next job released *tardy*.

Choose

$$\alpha = 2 - \varphi = \frac{3 - \sqrt{5}}{2} \approx 0.38197.$$

We claim that SLEEPY is $(1+\alpha)$-competitive. By considering two jobs of size 1 released at time 0 we see that SLEEPY is no better than $(1 + \alpha)$-competitive.

3. Analysis

We will show, by contradiction, that no set of jobs with optimal offline makespan 1 causes SLEEPY to have a makespan more than $1 + \alpha$. This suffices, since we can rescale any nonempty job set to have optimal offline makespan 1.

Without loss of generality, we identify jobs with integers $1, \ldots, n$ so that $C_1 \leqslant C_2 \leqslant \cdots \leqslant C_n$. Assume that $\varrho = \{(p_i, r_i)\}_{i=1}^{n}$ is a minimum size set of jobs with optimal offline makespan 1 which causes SLEEPY to have a makespan $C_n > 1 + \alpha$. Let A be the machine which runs job n and B be the other machine.

The remainder of the analysis consists of verifying a number of claims. Intuitively, what we want to show is: (1) the last three jobs to be released are the last three jobs completed by SLEEPY, (2) two of the last three jobs have processing time greater than α and the third has processing time greater than $1 - \alpha$, and (3) replacing the last three jobs with one job results in a smaller counterexample.

The following claim is almost identical to one proven by Chen and Vestjens [7] for LPT:

Claim 3.1. *Without loss of generality, there is no time period during which* SLEEPY *has both machines idle.*

Proof. Assume both machines are idle during (t_1, t_2) and that there is no time later than t_2 when both machines are idle. If there is a job with release time prior to t_2

then removing this job results in a set with fewer jobs, the same optimal offline cost, and the same cost for SLEEPY. If no job is released before t_2 then $\varrho' = \{(p_i/(1-t_2), (r_i - t_2)/(1-t_2))\}_{i=1}^{n}$ is a job set with optimal offline makespan 1, greater cost for SLEEPY, and no time period when SLEEPY has both machines idle. Note ϱ' is simply ϱ with all release times decreased by t_2 and all jobs rescaled so that the optimal offline cost remains 1. □

Claim 3.2. *The last job on machine B completes at time* $C_{n-1} < 1$.

Proof. Since there is no period when SLEEPY has both machines idle, the last job on machine B must complete at or after s_n and so $C_{n-1} \geqslant s_n$.

We now show that from the release time of the last tardy job until C_{n-1} the online algorithm will process at an average rate of $2 - \alpha$. Intuitively, if $C_{n-1} \geqslant 1$ then there would be too much processing remaining for the optimal offline algorithm to complete by time 1.

Let j be the last tardy job and k be the job running when the last tardy job is released. If there are no tardy jobs let $r_j = C_k = 0$. At time r_j the optimal offline algorithm may or may not have completed all jobs released before r_j. Let p (possibly 0) be the amount of processing remaining to be completed by the optimal offline algorithm at time r_j on jobs released before r_j.

Since ϱ is a minimum size counterexample, $C_k \leqslant (1+\alpha)(r_j + p)$. Note that between time r_j and C_n the processing done by SLEEPY is exactly the processing done by the optimal offline algorithm plus $C_k - r_j - p$.

It follows from the previous claim that there is a set of jobs $\sigma_1, \ldots, \sigma_\ell$ such that $s_{\sigma_1} \leqslant C_k < C_{\sigma_1}$, $s_{\sigma_i} \leqslant s_{\sigma_{i+1}}$, and $C_{\sigma_\ell} = C_{n-1}$.

Consider the time period (C_k, C_{σ_1}). During this period neither machine is waiting, since the last tardy job arrived prior to C_k. So, at least $(2-\alpha)(C_{\sigma_1} - C_k)$ processing is completed by SLEEPY. Similarly, during the time period $(C_{\sigma_i}, C_{\sigma_{i+1}})$ at least $(2-\alpha)(C_{\sigma_{i+1}} - C_{\sigma_i})$ processing is completed by SLEEPY for $i = 1, 2, \ldots, \ell - 1$. Therefore, during the period (C_k, C_{n-1}), SLEEPY completes at least $(2-\alpha)(C_{n-1} - C_k)$ processing.

Since the optimal offline algorithm can process at most $2(1 - r_j)$ after r_j,

$$2(C_k - r_j) + (2-\alpha)(C_{n-1} - C_k) + (C_n - C_{n-1}) - (C_k - r_j - p) \leqslant 2(1 - r_j).$$

Therefore,

$$(1-\alpha)(C_{n-1} - C_k) \leqslant 2 - r_j - p - C_n$$

and further

$$\begin{aligned} C_{n-1} &\leqslant \frac{2 - r_j - p - C_n}{1-\alpha} + C_k \\ &= (2-\alpha)(2 - r_j - p - C_n) + C_k \\ &< (2-\alpha)(1 - \alpha - r_j - p) + C_k \end{aligned}$$

$$= 1 - (2 - \alpha)(r_j + p) + C_k$$

$$\leqslant 1 - (2\alpha - 1)(r_j + p)$$

$$\leqslant 1. \qquad \square$$

The previous claim implies that $p_n \geqslant C_n - C_{n-1} > \alpha$.

Claim 3.3. $s_n > s_{n-1} + \alpha p_{n-1}$.

Proof. If $s_n \leqslant s_{n-1}$ then removing job $n-1$ results in a smaller set with no greater offline cost and no less cost for SLEEPY, a contradiction. From the definition of SLEEPY, $s_n \geqslant s_{n-1} + \alpha p_{n-1}$. Suppose that this is satisfied with equality. If $p_n > p_{n-1}$ then $r_n \geqslant s_{n-1}$. Therefore, we have

$$C_n = s_n + p_n = s_{n-1} + \alpha p_{n-1} + p_n \leqslant \alpha p_{n-1} + r_n + p_n \leqslant 1 + \alpha.$$

On the other hand, if $p_n \leqslant p_{n-1}$ then

$$C_n = s_n + p_n \leqslant s_{n-1} + \alpha p_{n-1} + p_{n-1} \leqslant C_{n-1} + \alpha p_{n-1} \leqslant 1 + \alpha. \qquad \square$$

Claim 3.4. $C_{n-2} = s_n$.

Proof. Since $s_n > s_{n-1} + \alpha p_{n-1}$, either $C_{n-2} = s_n$ or machine A is idle immediately before time s_n. In the later case, we have $s_n = r_n$ and therefore $C_n = s_n + p_n = r_n + p_n \leqslant 1$, which is a contradiction. $\square$

Claim 3.5. $p_{n-1} \geqslant C_{n-2} - s_{n-1} > \alpha$.

Proof. It is easily seen that $p_{n-1} \geqslant C_{n-2} - s_{n-1}$. If $C_{n-2} - s_{n-1} \leqslant \alpha$ then $p_{n-1} = (C_{n-1} - C_{n-2}) + (C_{n-2} - s_{n-1}) \leqslant (C_{n-1} - C_{n-2}) + \alpha$. Therefore $p_{n-1} - C_{n-1} + C_{n-2} \leqslant \alpha$ and further $r_n \geqslant s_{n-1}$. This means $C_n = s_{n-1} + p_{n-1} + p_n - (C_{n-1} - C_{n-2}) \leqslant r_n + p_n + \alpha \leqslant 1 + \alpha$, a contradiction. $\square$

Claim 3.6. $p_{n-2} > \alpha$.

Proof. If $p_{n-2} \leqslant \alpha$ then $p_{n-2} < p_n$. So, $r_n \geqslant s_{n-2}$. This means $C_n = s_{n-2} + p_{n-2} + p_n \leqslant r_n + \alpha + p_n \leqslant 1 + \alpha$, a contradiction. $\square$

Of the jobs $n-1$ and $n-2$, let j be the job which starts first and k be the other.

Claim 3.7. $p_j > 1 - \alpha$.

Proof. We have $p_j = C_j - s_j \geqslant C_{n-2} - s_j$. Certainly, $s_j + \alpha p_j \leqslant s_k$. So, $(1-\alpha)(C_{n-2} - s_j) \geqslant C_{n-2} - s_j - \alpha p_j \geqslant C_{n-2} - s_k$. Therefore,

$$\begin{aligned} p_j &\geqslant C_{n-2} - s_j \\ &\geqslant \frac{C_{n-2} - s_k}{1-\alpha} \\ &= (2-\alpha)(C_{n-2} - \max\{s_{n-1}, s_{n-2}\}) \\ &= (2-\alpha)\min\{C_{n-2} - s_{n-1}, p_{n-2}\} \\ &> (2-\alpha)\alpha \\ &= 1-\alpha. \qquad \square \end{aligned}$$

Claim 3.8. $s_j + \alpha p_j = s_k$.

Proof. No job other than n, $n-1$, and $n-2$ requires more than α processing time, since the optimal offline makespan is 1. For the same reason jobs k and n must go on the same machine in the optimal offline schedule. This means either k or n must be released before time $1-2\alpha$. Suppose job $i < n-2$ has $s_i \leqslant s_j$ and $C_i \geqslant s_j + \alpha p_j$. Then $s_i + \alpha p_i \leqslant s_j$ and $p_i \geqslant \alpha p_i + \alpha p_j \geqslant \alpha p_i + \alpha(1-\alpha)$. This implies that $p_i \geqslant \alpha$, a contradiction. Therefore, any job that starts before job j must complete before time $s_j + \alpha p_j$: Since $\alpha p_j > \alpha(1-\alpha) = 1-2\alpha$, we have $s_k = s_j + \alpha p_j$. $\square$

Claim 3.9. $\min\{r_n, r_{n-1}, r_{n-2}\} > \max_{i \leqslant n-3}\{s_i\}$.

Proof. Since jobs n, $n-1$, and $n-2$ are larger than all other jobs and start later, they must have been released after $\max_{i \leqslant n-3}\{s_i\}$. $\square$

Claim 3.10. $p_n + p_k - p_j \geqslant 0$.

Proof. We have

$$\begin{aligned} p_n + p_k - p_j &= p_n + C_k - s_k - (C_j - s_j) \\ &= p_n + C_k - C_j - \alpha p_j \\ &\geqslant p_n + C_k - C_j - \alpha \\ &= C_n - C_{n-1} + C_{n-1} - C_{n-2} + C_k - C_j - \alpha \\ &> C_{n-1} - C_{n-2} + C_k - C_j \\ &\geqslant 0. \qquad \square \end{aligned}$$

Let ϱ' be the set of jobs $\{(p_i, r_i)\}_{i=1}^{n-3} \cup \{(p_n + p_k - p_j, \min\{r_n, r_{n-1}, r_{n-2}\})\}$. This set is valid by the preceding claim.

Claim 3.11. *The optimal offline cost of ϱ' is no more than* $1 - p_j$.

Proof. Since we can assume that on each of the machines individually the optimal offline algorithm orders its jobs by release times, this set can be processed identically to ϱ for the first $n-3$ jobs. The new job $(p_n + p_k - p_j, \min\{r_n, r_{n-1}, r_{n-2}\})$ runs on the machine which would have run k. This schedule has cost no more than $1 - p_j$. □

Claim 3.12. SLEEPY*'s cost on* ϱ' *is at least* $C_n - (1+\alpha)p_j$.

Proof. The first $n-3$ jobs are served identically. The final job starts at time s_j. The makespan is at least $s_j + p_n + p_k - p_j = s_k + p_k + p_n - (1+\alpha)p_j \geqslant C_n - (1+\alpha)p_j$. □

If we rescale ϱ' to have optimal offline cost 1 then we have our contradiction (a smaller counterexample). We therefore have:

Theorem 3.1. SLEEPY *is* $(1+\alpha)$*-competitive for* $\alpha = 2 - \varphi$.

4. Randomized lower bounds

We show a randomized lower bound which improves on the previous result of Stougie and Vestjens [20, 21]. To do this, we use the von Neumann and Yao principle [23, 22]. The use of this principle to prove lower bounds on the competitive ratio is treated in Borodin and El-Yaniv [4]. However, none of the results given there apply to our situation, since they allow a more forgiving definition of competitiveness. We therefore need the following lemma:

Lemma 4.1. *Let* χ *be a distribution over a set of inputs* $\{\sigma_\alpha\}$ *indexed by* α. *Let the set of all deterministic algorithms* $\{A_\beta\}$ *for this set of inputs be indexed by* β. *If there exists a real number* ρ *such that for all* β

$$E_\chi\left[\frac{\mathrm{cost}_{A_\beta}(\sigma_\alpha)}{\mathrm{cost}(\sigma_\alpha)}\right] \geqslant \rho,$$

then no randomized algorithm has competitive ratio less than ρ.

Proof. Suppose for a contradiction that a randomized online algorithm $\mathscr{A}$ with competitive ratio less than ρ exists. $\mathscr{A}$ is a distribution ψ over the set of deterministic online algorithms $\{A_\beta\}$. We have

$$\begin{aligned}\rho &\leqslant \inf_\beta E_\chi\left[\frac{\mathrm{cost}_{A_\beta}(\sigma_\alpha)}{\mathrm{cost}(\sigma_\alpha)}\right] \leqslant E_\psi E_\chi\left[\frac{\mathrm{cost}_{A_\beta}(\sigma_\alpha)}{\mathrm{cost}(\sigma_\alpha)}\right],\\ &= E_\chi E_\psi\left[\frac{\mathrm{cost}_{A_\beta}(\sigma_\alpha)}{\mathrm{cost}(\sigma_\alpha)}\right] \leqslant \sup_\alpha E_\psi\left[\frac{\mathrm{cost}_{A_\beta}(\sigma_\alpha)}{\mathrm{cost}(\sigma_\alpha)}\right] = \sup_\alpha \frac{E_\psi[\mathrm{cost}_{A_\beta}(\sigma_\alpha)]}{\mathrm{cost}(\sigma_\alpha)}.\end{aligned}$$

The first inequality follows from the conditions of the lemma. The rest follow from basic facts about mathematical expectations. This implies that for any real $\varepsilon > 0$ there exists an α such that

$$E_\psi[\mathrm{cost}_{A_\beta}(\sigma_\alpha)] \geqslant (\rho - \varepsilon)\mathrm{cost}(\sigma_\alpha),$$

which is a contradiction. □

Theorem 4.1. *No randomized algorithm for $m \geqslant 2$ machines is ρ-competitive with ρ less than*

$$1 + \max_{0 \leqslant u < 1} \frac{u}{1 - 2\ln(1-u)} > 1.21207.$$

Proof. We use Lemma 4.1. Let $0 \leqslant u < 1$ be a real constant. We show a distribution over job sets such that the expectation of the competitive ratio of all deterministic algorithms is at least $1 + u/(1 - 2\ln(1-u))$. Let

$$q = \frac{1}{1 - 2\ln(1-u)}, \qquad p(y) = \frac{1}{(y-1)\ln(1-u)}.$$

We now define the distribution χ. In all cases, we give two jobs of size 1 at time 0. With probability q, we give no further jobs. With probability $1-q$, we pick y at random from the interval $[0, u]$ using the probability density $p(y)$ and release $m-1$ jobs of size $2-y$ at time y. Note that $q \geqslant 0$ and that $\int_0^u p(y)\,\mathrm{d}y = 1$, and therefore this is a valid distribution.

We now analyze the expected competitive ratio incurred by a deterministic online algorithm on this distribution. Let x be the time at which the algorithm would start the second of the two size one jobs, given that it never receives the jobs of size $2-y$. If the third job is not given, the algorithm's cost is $1+x$ while the optimal offline cost is 1. If the jobs of size $2-y$ are given at time $y \leqslant x$, then the algorithm's cost is at least 2 and this is also the optimal offline cost. If the jobs of size $2-y$ are given at time $y > x$, then the algorithm's cost is at least $3-y$ while the optimal offline cost is again 2. First consider $x \geqslant u$. In this case, the expected competitive ratio is

$$\begin{aligned} E_\chi\left[\frac{\mathrm{cost}_A(\sigma)}{\mathrm{cost}(\sigma)}\right] &\geqslant q(1+x) + (1-q)\int_0^u p(y)\,\mathrm{d}y, \\ &\geqslant \frac{1+u}{1-2\ln(1-u)} + \frac{-2\ln(1-u)}{1-2\ln(1-u)}\int_0^u \frac{\mathrm{d}y}{(y-1)\ln(1-u)}, \\ &= 1 + \frac{u}{1-2\ln(1-u)}. \end{aligned}$$

Now consider $x < u$. In this case we have

$$\begin{aligned} &E_\chi\left[\frac{\mathrm{cost}_A(\sigma)}{\mathrm{cost}(\sigma)}\right] \\ &\quad \geqslant q(1+x) + (1-q)\left(\int_0^x p(y)\,\mathrm{d}y + \int_x^u p(y)\frac{3-y}{2}\,\mathrm{d}y\right), \end{aligned}$$

$$= \frac{1+x}{1-2\ln(1-u)} + \frac{-2}{1-2\ln(1-u)}\left(\int_0^x \frac{dy}{y-1} + \int_x^u \frac{3-y}{2}\frac{dy}{y-1}\right),$$
$$= \frac{1+x}{1-2\ln(1-u)} + \frac{-2}{1-2\ln(1-u)}\left(\ln(u-1) - \frac{u}{2} + \frac{x}{2}\right),$$
$$= 1 + \frac{u}{1-2\ln(1-u)}.$$

Since the adversary may choose u, the theorem follows. Choosing $u = 0.575854$ yields a bound of at least 1.21207. □

One natural class of randomized online algorithms is the class of *barely random* algorithms [15]. A barely random online algorithm is one which is a distribution over a constant number of deterministic algorithms. Such algorithms are often easier to analyze than general randomized ones. We show a lower bound for any barely random algorithm which is a distribution over two deterministic algorithms:

Theorem 4.2. *No distribution over two deterministic online algorithms is ρ-competitive with ρ less than $\frac{5}{4}$.*

Proof. Suppose there is an algorithm which is $(\frac{5}{4}-\varepsilon)$-competitive for $\varepsilon > 0$. We give the algorithm two jobs of size 1 at time 0. With some probability p, the algorithm starts the second job at time t_1, while with probability $1-p$, it start it at time t_2. Without loss of generality, $t_1 < t_2$. First, we must have

$$1 + pt_1 + (1-p)t_2 < \tfrac{5}{4},$$

and therefore

$$p > \frac{4t_2 - 1}{4(t_2 - t_1)}.$$

We also conclude that $t_1 < \frac{1}{4}$. Now suppose we give the algorithm $m-1$ jobs of size $2 - t_1 - \delta$ at time $t_1 + \delta$. Then its competitive ratio is at least

$$\frac{p(3-t_1-\delta) + (1-p)2}{2} = \frac{1-t_1-\delta}{2}p + 1 > \frac{(1-t_1-\delta)(4t_2-1)}{8(t_2-t_1)} + 1 < \tfrac{5}{4},$$

and therefore

$$t_2 < \frac{1-3t_1-\delta}{2-4t_1-4\delta}.$$

Since this is true for all $\delta > 0$ we have

$$t_2 < \frac{1-3t_1}{2-4t_1}.$$

If instead we give $m-1$ jobs of size $2 - t_2 - \delta$ at time $t_2 + \delta$, then the competitive ratio is at least

$$\frac{3-t_2-\delta}{2} > \frac{5-9t_1-2\delta+4t_1\delta}{4-8t_1}.$$

Again, this holds for all $\delta > 0$ and so we have

$$\frac{5 - 9t_1}{4 - 8t_1} \leqslant \frac{5}{4} - \varepsilon,$$

which is impossible for $0 \leqslant t_1 < \frac{1}{4}$. □

This is interesting in that it shows that to be $\frac{5}{4}$-competitive, a distribution over 2 deterministic algorithms should start the initial jobs immediately or delay the start of one of the jobs by $\frac{1}{2}$ the processing time, both with equal probability. It suggests the following algorithm for two machines, which we call HALF SLEEPY:

1. With probability $\frac{1}{2}$ run LPT.
2. With probability $\frac{1}{2}$ run SLEEPY with $\alpha = \frac{1}{2}$.

We leave as an open question the competitive ratio of HALF SLEEPY.

5. Conclusions

We have shown a deterministic online algorithm for scheduling two machines with release times with the best possible competitive ratio. Further, we have improved the lower bound on the competitive ratio of randomized online algorithms for this problem.

We feel there are a number of interesting and related open questions. First, is there a way to generalize SLEEPY to an arbitrary number of machines and have a competitive ratio smaller than LPT? Or the even stronger result, for a fixed value of $\varepsilon > 0$ is there an algorithm which is $(\frac{3}{2} - \varepsilon)$-competitive for all m? Second, does randomization actually reduce the competitive ratio? It seems like randomization should help to reduce the competitive ratio a great deal. However, the best known randomized algorithms are actually deterministic (SLEEPY for $m = 2$ and LPT for $m > 2$). We have proposed a randomized algorithm HALF SLEEPY, but so far have not been successful in analyzing it. Third, how much does the competitive ratio decrease if restarts are allowed? In many real-world situations a job can be killed and restarted later with only the loss of processing already completed.

Acknowledgements

The authors would like to thank Gerhard Woeginger for suggesting the problem.

References

[1] S. Albers, Better bounds for online scheduling. In Proc. 29th ACM Symp. on Theory of Computing, 1997, pp. 130–139.
[2] Y. Bartal, A. Fiat, H. Karloff, R. Vohra, New algorithms for an ancient scheduling problem, J. Comput. System Sci. 51 (3) (1995) 359–366.
[3] Y. Bartal, H. Karloff, Y. Rabani, A better lower bound for on-line scheduling, Informat. Process. Lett. 50 (3) (1994) 113–116.

[4] A. Borodin, R. El-Yaniv, Online Computation and Competitive Analysis, Cambridge University Press, Cambridge, 1998.
[5] B. Chen, A. van Vliet, G. Woeginger, A lower bound for randomized on-line scheduling algorithms, Informat. Process. Lett. 51 (5) (1994) 219–222.
[6] B. Chen, A. van Vliet, G. Woeginger, New lower and upper bounds for on-line scheduling, Operat. Res. Lett. 16 (4) (1994) 221–230.
[7] B. Chen, A. Vestjens, Scheduling on identical machines. How good is LPT in an online setting?, Operat. Res. Lett. 21 (4) (1997) 165–169.
[8] U. Faigle, W. Kern, G. Turàn, On the performance of on-line algorithms for partition problems, Acta Cybernetica 9 (2) (1989) 107–119.
[9] G. Galambos, G. Woeginger, An online scheduling heuristic with better worst case ratio than Graham's list scheduling, SIAM J. Comput. 22 (2) (1993) 349–355.
[10] R.L. Graham, Bounds for certain multiprocessing anomalies, Bell Systems Technic. J. 45 (1966) 1563–1581.
[11] L. Hall, D. Shmoys, Approximation schemes for constrained scheduling problems, In Proc. 30th IEEE Symp. on Foundations of Computer Science, 1989, pp. 134–139.
[12] D. Karger, S. Phillips, E. Torng, A better algorithm for an ancient scheduling problem, J Algorithms 20 (2) (1996) 400–430.
[13] A. Karlin, M. Manasse, L. Rudolph, D. Sleator, Competitive snoopy caching, Algorithmica 3 (1) (1988) 79–119.
[14] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, D.B. Shmoys, Sequencing and scheduling: Algorithms and complexity. In: S.C. Graves, A.H.G.R. Kan, P. Zipkin (Eds.), Handbooks in Operations Research and Management Science, Vol. 4: Logistics of Production and Inventory, North-Holland, 1993, pp. 445–552.
[15] N. Reingold, J. Westbrook, D. Sleator, Randomized competitive algorithms for the list update problem, Algorithmica 11 (1) (1994) 15–32.
[16] S.S. Seiden, Randomized algorithms for that ancient scheduling problem, Proc. 5th Workshop on Algorithms and Data Structures, 1997, pp. 210–223.
[17] J. Sgall, A lower bound for randomized on-line multiprocessor scheduling, Informat. Process. Lett. 63 (1) (1997) 51–55.
[18] D. Shmoys, J. Wein, D. Williamson, Scheduling parallel machines on-line, Proc. 32nd Symp. on Foundations of Computer Science, 1991, pp. 131–140.
[19] D. Sleator, R. Tarjan, Amortized efficiency of list update and paging rules, Communications of the ACM 28 (2) (1985) 202–208.
[20] L. Stougie, A.P.A. Vestjens, Randomized on-line scheduling: How low can't you go? Manuscript, 1997.
[21] A.P.A. Vestjens, On-line Machine Scheduling, Ph.D. Thesis, Eindhoven University of Technology, The Netherlands, 1997.
[22] J. Von Neumann, O. Morgenstern, Theory of games and economic behavior, 1st edition, Princeton University Press, Princeton, 1944.
[23] A.C. Yao, Probabilistic computations: Toward a unified measure of complexity, In Proc. 18th IEEE Symp. on Foundations of Computer Science, 1977, pp. 222–227.