A lower bound on deterministic online algorithms for scheduling on related machines without preemption

Tomáš Ebenlendr* Jiří Sgall†

December 2, 2011

Abstract

We prove a new lower bound of 2.564 on deterministic online algorithms for makespan scheduling on related machines (without preemptions). Previous lower bound was 2.438 by Berman et al. We use an analytical bound on maximal frequency of scheduling jobs instead of the combinatorial bound obtained by computer based search through the graph of possible states of an algorithm in the previous work.

1 Introduction

We consider *one-by-one* online scheduling on uniformly related machines. The speed of machine M_i is denoted s_i. Each job is characterized by its processing time p_j takes p_j/s_i time to process on M_i. No preemptions are allowed, i.e., once the job is started it cannot be interrupted and the machine is busy with this job until the job is processed. The objective is to minimize the *makespan* (also called the length of the schedule, or the maximal completion time). The online algorithm sees only the next job from the input sequence and it has to schedule this job before it is given the following job. Note that in this model it is not necessary to specify the starting times of jobs, as any schedule can be trivially converted to the schedule without idle time (gaps), while not increasing the makespan. (Accordingly, this model is often considered as a variant of load balancing.)

We prove a new lower bound of 2.564 for the above-described problem, i.e., for deterministic online algorithms for makespan scheduling on related machines without preemptions. The previous lower bound was 2.438 by Berman et al [5]. They use combinatorial approach with computer based search through the graph of possible states of an algorithm. In contrast, we use an analytical bound on maximal frequency of scheduling jobs.

Our lower bound is based on an instance where both the machine speeds and the processing times are a geometric sequence of machines, with both sequences having the same common ratio, similarly as in [5, 11]. In the previous bounds for similar problems one usually argues about the total amount of work done by the machines. In contrast, our bound is based on reasoning about the number of jobs scheduled and the frequency of scheduling jobs on every machine. First, we consider how the algorithm behaves on one of the machines and we upper bound the frequency of scheduling a job on this machine. This bound is a function of the

*Institute of Mathematics, AS CR, Žitná 25, CZ-11567 Praha 1, Czech Republic. Email: ebik@math.cas.cz.

†Dept. of Applied Mathematics, Faculty of Mathematics and Physics, Charles University, Malostranské nám. 25, CZ-11800 Praha 1, Czech Republic. Email: sgall@kam.mff.cuni.cz.

competitive ratio, the common ratio of the geometric sequence, and the speed of the machine. Then we take the sum of these bounds on frequencies over all machines. Any online algorithm has to schedule one job in one step, thus this sum has to be at least 1. Finally, we let the common ratio of the geometric sequence to approach 1, and obtain our lower bound. This yields a certain inequality for the competitive ratio which we solve numerically.

Related work

Naturally, the lower bounds need to be compared to the existing algorithms. The first constant-competitive algorithm for non-preemptive scheduling on related machines was developed in [1]. The currently best algorithms are $3 + \sqrt{8} \approx 5.828$ competitive deterministic and 4.311 competitive randomized one [5]. For an alternative very nice presentation see [3]. All these algorithms use doubling, i.e., strategies that estimate the optimal makespan by a geometric sequence. While this is a standard technique for obtaining a constant competitive ratio, it would be surprising if it led to optimal algorithms. The lower bound for randomized algorithms is 2, see [11]. Thus, both in the deterministic and randomized cases, significant gaps remain.

For a small number of machines the best known algorithm is the the greedy List Scheduling. Here List Scheduling is defined so that the next job is always scheduled so that it will finish as early as possible. For many machines it is far from optimal, its competitive ratio is $\Theta(\log n)$, see [1]. The exact competitive ratio for $m = 2$ is ϕ and for $3 \leq m \leq 6$ it is equal to $1 + \sqrt{(m-1)/2}$ [7]; moreover for $m = 2, 3$ it can be checked easily that there is no better deterministic algorithm. For $m = 2$ it is possible even to give the exact optimal ratio for any speed combination, see [10]. Even for three machines it is not known exactly for which speed combinations is List Scheduling optimal, some recent progress is reported in [6, 12]. Another special case when some partial results about optimality of List Scheduling are known is the case when $m - 1$ machines have the same speed, see e.g. [12, 13].

The previous lower bound of 2.438 works for $m = 9$; for a smaller number of machines, no lower bound is known, except for the bound of 2 that follows from the analysis of List Scheduling for $m = 3$.

Interestingly, for the related problem where preemptions are allowed, we are able to provide an optimal online algorithm for any combination of the speeds and its competitive ratio is between 2.112 and $e \approx 2.718$, see [9, 8]. Similar results seem to be out of reach for non-preemptive scheduling, as the combinatorial structure is much more difficult and the value of the optimum is NP-hard to compute, while for the preemptive scheduling it is computable, in fact given by an easy formula.

The problem of non-preemptive scheduling can be formulated in the language of online load balancing as the case where the jobs are permanent and the load is their only parameter corresponding to our processing time. Consequently, there are many results on load balancing that extend the basic results on online scheduling in a different direction, see e.g. [2].

Notations

We number the machines as well as the jobs from 0 (to obtain simpler formulas). Thus we have machines $M_0, M_1, \ldots, M_{m-1}$ and jobs $\mathcal{J} = (J_0, J_1, \ldots, J_{n-1})$. We use $\mathcal{J}[j] = (J_0, J_1, \ldots, J_j)$ to denote the input sequence of jobs cut off after J_j.

Let $\mathcal{J}_i$ be the set of jobs scheduled on machine M_i. The completion time of the machine is then simply the sum of processing times of the jobs scheduled to the machine divided by its speed: $C_i = \frac{1}{s_i} \sum_{j:J_j \in \mathcal{J}_i} p_j$. We compare the maximum completion time in the output of the algorithm with maximum completion time of the optimal schedule. The completion time of job j is defined as the completion time of the machine where it is scheduled just after scheduling it, i.e., $\frac{1}{s_i} \sum_{k:k<j, J_k \in \mathcal{J}_i} p_j$.

2 Lower bound

Our lower bound is proved by an instance with a geometric sequence of machines, $s_i = \alpha^{-i}$, and a geometric sequence of jobs, $p_i = \alpha^i$, for some $\alpha > 1$. Both sequences have the same length, i.e., $n = m$. The optimal schedule after step t is to schedule the jobs on the machines in the reverse order, i.e., the J_j on machine M_{t-j}. The optimal makespan is thus equal to the size of the largest job, $C^*_{\max}(\mathcal{J}[t]) = p_t = \alpha^t$.

To achieve the competitive ratio of R, the algorithm has to complete the job t before time $R \cdot C^*_{\max}(\mathcal{J}[t])$. It follows immediately that it cannot schedule any job at any machine with speed below $1/R$. Furthermore, if the speed of a machine is only slightly above $1/R$, the jobs cannot be scheduled on it very often. Intuitively, the faster machines can schedule a job more frequently. We calculate the maximal frequency of scheduling a job for each machine separately, depending on its speed and R. The lower bound will follow from the fact that the sum of the frequencies has to be at least 1 so that the algorithm schedules all the jobs.

Following this scheme of the proof has some technical difficulties. In particular, the notion of frequency is not clear: The algorithm may schedule nothing on a machine for some time and then several jobs in a row. We need to think in a certain amortized way. Instead of formalizing the notion of amortized frequency, we formulate the bounds in terms of the number of jobs.

The following main lemma gives the bound for a single machine. The number t_i can be interpreted as the highest possible amortized frequency of scheduling a job to machine M_i with respect to the claimed competitive ratio R.

Lemma 1 *Let* A *be an R-competitive algorithm. Consider the instance described above. Let*

$$t_i \quad = \quad \log_\alpha \frac{R}{R - \alpha^i} \qquad \text{for } s_i = \alpha^{-i} > R^{-1} \tag{1}$$

Then, for any fixed $\alpha > 1$ and n, the algorithm A *schedules at most $\frac{n}{t_i} + \lceil R \rceil$ jobs from the input sequence on machine M_i. Moreover the algorithm schedules at most one job on the machine with speed equal to $1/R$ (if there is any) and no job on any slower machine.*

Proof: If $s_i < 1/R$, then no job can be scheduled the machine M_i, since if the sequence would end now, the optimal makespan would be equal to the size of the last job on input, i.e, $C^*_{\max}(\mathcal{J}[t]) = p_t$. Moreover if $s_i = 1/R$ then only one job can be scheduled on M_i: The same argument now shows that no job is scheduled on M_i before scheduling any job. Thus we assume $s_i > 1/R$ from now on.

Let $c_1, c_2, \ldots, c_{n_i}$ be the completion times of the jobs in $\mathcal{J}_i$ (i.e., those scheduled on the machine M_i). First we claim that for all $k = 2, \ldots, n_i$,

$$\frac{c_{k+1}}{c_k} \geq \frac{R}{R - \alpha^i}. \tag{2}$$

The processing time of the job completing at c_{k+1} equals $s_i(c_{k+1} - c_k)$, thus the optimal makespan for an instance ending by this job is $s_i(c_{k+1} - c_k)$ as well. Since the algorithm A is R competitive, we have $c_{k+1} \leq Rs_i(c_{k+1} - c_k) = R\alpha^{-i}(c_{k+1} - c_k)$ and the claim (2) follows.

Let k_0 be the first index such that $c_{k_0} \geq R$. Note that $k_0 \leq \lceil R \rceil$ as all jobs have processing times at least 1 and the machine speeds are at most 1.

Considering the instance with n jobs, we have $c_{n_i} \leq R \cdot \alpha^n$. Applying the claim (2) for $n_i - k_0$ pairs of adjacent completion times implies that

$$\alpha^n = \frac{R \cdot \alpha^n}{R} \geq \frac{c_{n_i}}{c_{k_0}} = \frac{c_{n_i}}{c_{n_i-1}} \cdot \cdots \cdot \frac{c_{k_0+1}}{c_{k_0}} \geq \left(\frac{R}{R - \alpha^i}\right)^{n_i - k_0}$$

and, after taking logarithm with base α,

$$n \geq (n_i - k_0) \log_\alpha \frac{R}{R - \alpha^i} = (n_i - k_0) t_i.$$

The lemma now follows by recalling that $k_0 \leq \lceil R \rceil$. □

Let us define

$$\begin{aligned} f(R, x) &= \frac{\ln(R)}{-\ln(1 - R^{-x})} \quad \text{and} \\ F(R) &= \int_0^1 f(R, x)\, dx = \int_0^1 \frac{\ln(R)}{-\ln(1 - R^{-x})}\, dx\,. \end{aligned}$$

The function $f(R, x)$ is an increasing function of both R and x (on positive real numbers). Thus $F(R)$ is increasing as well. Let R^* be the solution of the equation $F(R^*) = 1$.

Theorem 2 *For any R-competitive deterministic algorithm for nonpreemptive scheduling on related machines, the following inequality holds:*

$$1 \leq F(R) = \int_0^1 \frac{\ln(R)}{-\ln(1 - R^{-x})}\, dx\,. \tag{3}$$

This gives $R > R^* > 2.564$.

Proof: Let n_i be the number of jobs scheduled on the machine M_i at the end of the sequence. The algorithm has to schedule all jobs, thus Lemma 1 implies

$$n = \sum_{i=0}^{n-1} n_i = \sum_{i=0}^{\lfloor \log_\alpha R \rfloor} n_i \leq \lceil R \rceil \cdot \lfloor \log_\alpha R \rfloor + \sum_{i=0}^{\lceil \log_\alpha R \rceil - 1} \frac{n}{t_i}\,.$$

(The change of the summation bound is an artifact of the subtlety of $i = \log_\alpha R$: There t_i is not defined but $n_i \leq 1$.) We can set n arbitrarily large, so that the term $\lceil R \rceil \cdot \lfloor \log_\alpha R \rfloor$ is negligible. Thus, for any $\varepsilon > 0$, we get:

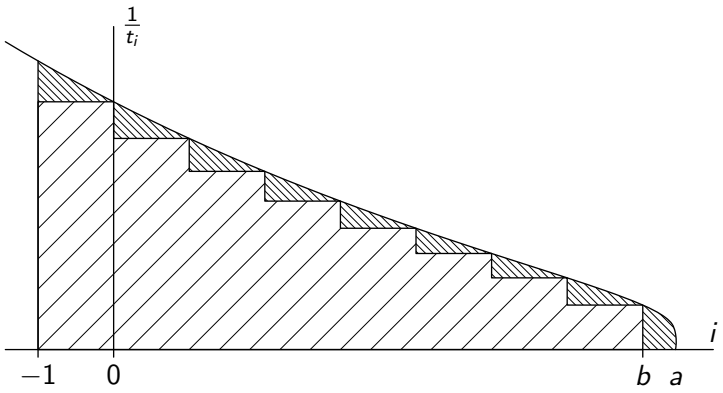

Figure 1: The labels on the horizontal axis are $a = \log_\alpha R$ and $b = \lceil \log_\alpha R \rceil - 1$. The sparsely hatched region shows the area of the sum in (4). The densely hatched region shows the additional area of the integral in (5).

$$
\begin{aligned}
1-\varepsilon \quad &\leq \quad \sum_{i=0}^{\lceil \log_\alpha R \rceil - 1} \frac{1}{t_i} \;=\; \sum_{i=0}^{\lceil \log_\alpha R \rceil - 1} \frac{\ln \alpha}{-\ln(1-\alpha^i R^{-1})} && (4)\\
&\leq \quad \int_{-1}^{\log_\alpha R} \frac{\ln \alpha}{-\ln(1-\alpha^i R^{-1})}\, di && (5)\\
&= \quad \int_{-\frac{\ln \alpha}{\ln R}}^{1} \frac{\ln R}{-\ln(1-R^{y-1})}\, dy && (6)\\
&= \quad \int_0^{1+\frac{\ln \alpha}{\ln R}} f(R,x)\, dx && (7)\\
& && (8)\\
&\xrightarrow{\alpha \to 1} \quad \int_0^1 f(R,x)\, dx && (9)\\
&= \quad F(R) \;\; . && (10)
\end{aligned}
$$

In (5) we simply bound the sum by the appropriate integral. We use the fact that the function in the sum can be viewed as a continuous and decreasing function of i, see Figure 1. We substitute $i = y \log_\alpha R = y \frac{\ln R}{\ln \alpha}$ to get (6) and $x = 1 - y$ to get (7). Since $f(R,x)$ is an increasing function of both R and x, the limit in (9) holds.

The monotonicity of f also makes it easy to evaluate the integration numerically and get the threshold of $R^* \approx 2.5649877$. □

A natural question is the size of the instance we need in our lower bound. Clearly, to achive a bound close to R^*, we need α close to 1 and a large n. However, our limit argument gives little intuition about the size of n. In the rest of this section we show that to prove a lower bound of $R' = R^* - \delta$, the size of the instance is $O(1/\delta^2)$.

First we need to calculate $1 - F(R')$. While we do not have a closed form for $F(R)$, it

can be verified that

$$\begin{aligned} \frac{\partial}{\partial R} f(R,x) &= \frac{x}{-R \cdot \ln(1-R^{-x})} \quad \text{and thus} \\ \frac{d}{dR} F(R) &= \int_0^1 \frac{\partial}{\partial R} f(R,x)\, dx = \frac{1}{-R \cdot \ln(1-R^{-1})}\,. \end{aligned}$$

So the derivative of $F(R)$ for R close to R^* is bounded and bounded away from zero, thus $1 - F(R') = \Theta(\delta)$.

Let $\varepsilon = (1 - F(R'))/2 = \Theta(\delta)$. We set α and n so that the errors in the two limits $\alpha \to 1$ and $\varepsilon \to 0$ are each bounded by ε.

First we choose $\alpha = 1 + \tau$ such that

$$\int_1^{1+\frac{\ln\alpha}{\ln(R')}} f(R',x)\, dx \le \varepsilon.$$

We have $\ln\alpha/\ln(R') = \Theta(\tau)$. Since $f(R,x)$ is increasing in R and x, we can bound its values by some constant M (arbitrarily close to $f(R^*,1)$, which is positive), and we have

$$\int_1^{1+\frac{\ln\alpha}{\ln(R')}} f(R',x)\, dx \le \frac{\ln\alpha}{\ln(R')} \cdot M = \Theta(\tau).$$

Thus it is sufficient to choose $\tau = \Theta(\varepsilon) = \Theta(\delta)$ and $\alpha = 1 + \Theta(\delta)$.

Finally, we need to choose n large so that

$$\frac{\lceil R' \rceil \lfloor \log_\alpha(R') \rfloor}{n} \le \varepsilon.$$

We have $\lceil R' \rceil \lfloor \log_\alpha(R') \rfloor = \Theta(1/\ln\alpha) = \Theta(1/\tau) = \Theta(1/\delta)$. Thus it is sufficient to choose $n = \Theta(1/\delta)/\varepsilon = \Theta(1/\delta^2)$.

Numerical calculation shows that for $\delta < 1/10$ it is sufficient to take $\varepsilon = \delta/3$, $\tau = \varepsilon/3$, and $n = 4/(\varepsilon\tau)$, overall this gives $n = 36/\delta^2$.

3 Conclusions

We have been able to improve the lower bound for non-preemptive online scheduling on related machines. The advantage of the new lower bound is that it provides a clean analytical argument. On the other hand, it seems that the limit case with many machines may not be the hardest one. For a fixed small number of machines, we assume that the combinatorial structure of the problem could lead to new lower bounds. This would probably need some combination of our analytical approach and the enumerative techniques from [5].

Our techniques cannot be used for lower bounds against the randomized algorithms, the best lower bound in this case remains at 2.

Of course, a challenge in this area is to design new algorithms, perhaps not based on the doubling techniques used so far.

Acknowledgments

Partially supported by Inst. for Theor. Comp. Sci., Prague (project 1M0545 of MŠMT ČR), grant IAA100190902 of GA AV ČR, and grant 166610 of GA UK. We are grateful to anonymous reviewers for helpful comments.

References

[1] James Aspnes, Yossi Azar, Amos Fiat, Serge Plotkin, and Orli Waarts. On-line load balancing with applications to machine scheduling and virtual circuit routing. *J. ACM*, 44:486–504, 1997.

[2] Y. Azar. On-line load balancing. In Amos Fiat and Gergard J. Woeginger, editors, *Online Algorithms: The State of the Art*, pages 178–195. Springer, 1998.

[3] Amotz Bar-Noy, Ari Freund, and Joseph Naor. New algorithms for related machines with temporary jobs. *J. Sched.*, 3:259–272, 2000.

[4] Piotr Berman, Moses Charikar, and Marek Karpinski. On-line load balancing for related machines. In *Proc. 5th Workshop on Algorithms and Data Structures (WADS)*, volume 1272 of *Lecture Notes in Comput. Sci.*, pages 116–125. Springer, 1997.

[5] Piotr Berman, Moses Charikar, and Marek Karpinski. On-line load balancing for related machines. *J. Algorithms*, 35:108–121, 2000.

[6] Sheng-Yi Cai and Qi-Fan Yang. Online scheduling on three uniform machines. *J. Comb. Optim.*, 2011. to appear.

[7] Yookun Cho and Sartaj Sahni. Bounds for list schedules on uniform processors. *SIAM J. Comput.*, 9:91–103, 1980.

[8] Tomáš Ebenlendr. *Combinatorial algorithms for online problems: Semi-online scheduling on related machines*. PhD thesis, Charles University, Prague, 2011.

[9] Tomáš Ebenlendr, Wojciech Jawor, and Jiří Sgall. Preemptive online scheduling: Optimal algorithms for all speeds. *Algorithmica*, 53:504–522, 2009.

[10] Leah Epstein, John Noga, Steven S. Seiden, Jiří Sgall, and Gerhard J. Woeginger. Randomized on-line scheduling for two uniform machines. *J. Sched.*, 4:71–92, 2001.

[11] Leah Epstein and Jiří Sgall. A lower bound for on-line scheduling on uniformly related machines. *Oper. Res. Lett.*, 26:17–22, 2000.

[12] Fangqiu Han, Zhiyi Tan, and Yang Yang. On the optimality of list scheduling for online uniform machines scheduling. *Optim. Lett.*, 2011. to appear.

[13] Antoine Musitelli and Jean-Marc Nicoletti. Competitive ratio of list scheduling on uniform machines and randomized heuristics. *J. Sched.*, 14:89–101, 2011.