Theory of Computing Systems manuscript No.
(will be inserted by the editor)

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

Tomáš Ebenlendr · Jiří Sgall

Received: date / Accepted: date

Abstract We consider one-by-one online scheduling on uniformly related machines. The input is a sequence of machines with different speeds and a sequence of jobs with different processing times. The output is a schedule which assigns the jobs to the machines; the completion time of a machine is the sum of the processing times of jobs assigned to it divided by its speed. The objective is to minimize the maximal completion time. The jobs arrive one by one and each has to be assigned to one machine immediately and irrevocably without the knowledge of the future jobs. We prove a new lower bound of 2.564 on the competitive ratio of deterministic online algorithms for this problem, improving the previous lower bound of 2.438.

Keywords Online algorithms · Scheduling · Makespan · Uniformly related machines · Lower bounds

1 Introduction

We prove a new lower bound for online scheduling on uniformly related machines. This is one of the simplest and most fundamental scheduling problems, yet it remains widely open and there have been no recent progress on it.

The instance of this problem consists of a sequence of machines with possibly different speeds and a sequence of jobs specified by their processing times.

Partially supported by the Center of Excellence – Inst. for Theor. Comp. Sci., Prague (project P202/12/G061 of GA ČR) and grant IAA100190902 of GA AV ČR.

T. Ebenlendr
Institute of Mathematics, AS CR, Žitná 25, CZ-11567 Praha 1, Czech Republic.
E-mail: ebik@math.cas.cz

J. Sgall
Computer Science Inst. of Charles University, Faculty of Mathematics and Physics, Malostranské nám. 25, CZ-11800 Praha 1, Czech Republic.
E-mail: sgall@iuuk.mff.cuni.cz

A *schedule* assigns each job to one of the machines; the time needed to process a job is equal to its processing time divided by the speed of the machine where it is assigned. The objective is to minimize the *makespan* (also called the length of the schedule, or the maximal completion time). Usually a schedule also needs to specify the timing of each job (its starting and completion times) so that the jobs on each machine do not overlap. Due to the simplicity of the problem we consider, this is not necessary and it is sufficient to specify the assignment to the machines, silently assuming that each job is started as soon as all the previous jobs on its machine are processed. Instead of calculating the completion times individually for each job, we can calculate the completion time of each machine as the total processing time of the jobs allocated to it divided by the speed of the machine; the makespan is then the maximum of the completion times over all machines.

In the online version of the problem, jobs appear online *one-by-one*. When a job appears, an online algorithm has to make an irrevocable decision and assign the job to a machine. This decision is permanent and made without the knowledge of the future jobs; the algorithm is not even aware of whether any future jobs exist or not. An online algorithm is *R-competitive* if for each instance it produces a schedule with makespan at most R times the optimal makespan.

We prove a new lower bound of 2.564 for the above-described problem, i.e., we show that no deterministic online algorithms for makespan scheduling on uniformly related machines is 2.564-competitive. The previously best lower bound was 2.438 given by Berman et al [4,5]. They use a combinatorial approach with a computer search through the graph of discretized possible states of an online algorithm. In contrast, we use an analytical bound obtained by reasoning about the maximal number of scheduled jobs on each machine.

Our lower bound is based on an instance where both the machine speeds and the processing times are a geometric sequence, with both sequences having the same common ratio, similarly as in [5,11]. The jobs arrive in the order of increasing processing times. Since from the perspective of the online algorithm, each coming job may be the last one in the sequence, it is necessary to assign all jobs to sufficiently fast machines. As the sequence gets longer, it turns out that the optimum would instead schedule the small initial jobs on slower machines. Intuitively, the now small jobs block the precious capacity of the fast machines. To obtain a lower bound, it is necessary to quantify this intuition. 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 on every machine. First, we consider how the algorithm behaves on one of the machines and we upper bound the number of jobs scheduled on this machine. This bound is a function of the competitive ratio, the common ratio of the geometric sequence, the speed of the machine, and the number of jobs in the sequence. It is linear in the number of jobs (disregarding a negligible additive term), thus the coefficient in this linear dependence can be interpreted as the maximal frequency of scheduling a job on that 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 uniformly related machines was developed in [1]. The currently best deterministic algorithms achieve competitive ratio $3+\sqrt{8} \approx 5.828$ and randomized algorithms are 4.311-competitive [5]. For an alternative very nice presentation see [3]. All these algorithms use doubling, i.e., strategies that work with some estimate of the optimal makespan and when it turns out that the estimate is too low, it is multiplied by 2 or some other constant. 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 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]. In contrast, even for three machines it is not known exactly for which speed combinations is List Scheduling optimal (even though for the worst possible combination of speeds we know that it is 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. [12,14].

The previous lower bound of 2.438 works for $m = 9$; for a smaller number of machines, no lower bound was known, except for the bound of 2 that follows from the analysis of List Scheduling for $m = 3$. Subsequent to our work and partially building on its ideas, new lower bounds for a small number of machines ($m = 4, \ldots, 11$) were found in [13].

It is interesting to compare our results to the related problem of preemptive scheduling on uniformly related machines. In that problem, each job can be divided into several pieces that may be scheduled on different machines so that the corresponding time slots are non-overlapping; the time slots may also be non-consecutive, so that at some times a partially completed job is not being processed at all. Contrary to our (non-preemptive) problem, with preemption it is necessary to specify the schedule completely, including the timing information. In the online version, for each job we have to specify

its schedule completely before the next job is revealed. Interestingly, for this preemptive problem, it is possible to give 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 rich 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 also 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. The arriving jobs are assigned to the machines and the goal is to minimize the maximal load of a machine, where a load of a machine is the sum of the loads of jobs divided by the speed of that machine. 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 $J_0, J_1, \ldots, J_{n-1}$. The speed of machine M_i is denoted by s_i. The processing time of job J_j is denoted by p_j; thus the job takes time p_j/s_i to be processed 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; in particular each job is assigned to a single machine. 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: $C_i = \frac{1}{s_i} \sum_{j:J_j \in \mathcal{J}_i} p_j$. The makespan is then the maximal completion time over all machines and we compare the makespan in the output of the algorithm with the makespan of an optimal schedule.

We use $\mathcal{J} = (J_0, J_1, \ldots, J_{n-1})$ and $C^*_{\max}(\mathcal{J})$ to denote the input sequence of jobs and its optimal makespan. Furthermore, let $\mathcal{J}[j] = (J_0, J_1, \ldots, J_j)$ and $C^*_{\max}(\mathcal{J}[j])$ denote the input sequence of jobs cut off after J_j and the optimal makespan of this initial segment of the input sequence. The initial segments are important, as the online algorithm needs to guarantee that it is competitive even if the sequence ends after the current job. For the same reason it is useful to consider the completion time of job J_j, defined as the completion time of the machine where J_j is scheduled just after scheduling it, i.e., $\frac{1}{s_i} \sum_{k:k \le j, J_k \in \mathcal{J}_i} p_j$.

2 Lower bound

Our lower bound is proved by the following instance. Both sequences of machines and jobs have the same length, i.e., $n = m$. The speeds of machines form

a geometric sequence, namely we set $s_i = \alpha^{-i}$, for some $\alpha > 1$. The processing times of jobs also form a geometric sequence, namely we set $p_i = \alpha^i$. The optimal schedule after step t, i.e., for $\mathcal{J}[t]$, 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 processing time of the largest job that runs on machine M_0 with speed 1, so $C^*_{\max}(\mathcal{J}[t]) = p_t = \alpha^t$.

To achieve the competitive ratio of R on the initial segment $\mathcal{J}[t]$, the algorithm has to complete the job J_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 for each machine separately the maximal number of jobs that can be scheduled on it, depending on its speed, R, and n (the number of jobs). The lower bound will follow from the fact that the sum of these bounds needs to be at least n so that the algorithm schedules all the jobs. The following main lemma gives these bounds for a single machine. The coefficient $1/t_i$ in the bound on the number of jobs can also be interpreted as the highest possible frequency of scheduling a job to machine M_i with respect to the claimed competitive ratio R.

Lemma 2.1 *Let* A *be an R-competitive algorithm. Consider the instance described above. Let*

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

Then, for any fixed $\alpha > 1$ and n, the algorithm A *schedules at most $\lceil R \rceil + \frac{n}{t_i}$ 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. To see this, observe that if the sequence ended at that moment, the optimal makespan would be equal to the processing time of the last job on input, i.e, $C^*_{\max}(\mathcal{J}[t]) = p_t$ for the current t, while the makespan of the algorithm would be at least p_t/s_i, which is strictly larger than $R \cdot p_t$. Moreover, if $s_i = 1/R$, then only a single job can be scheduled on M_i: After scheduling the second job, the makespan of the algorithm would be strictly larger than $p_t/s_i = R \cdot p_t$, thus the competitive ratio R would be violated again. 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). If $n_i \leq \lceil R \rceil$, the lemma holds. Otherwise let k_0 be the first index such that $c_{k_0} \geq R$. Note that all jobs have processing times at least 1 and the machine speeds are at most 1, thus k_0 exists and $k_0 \leq \lceil R \rceil$.

First we claim that for all $k = 2, \ldots, n_i$,

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

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

Considering the instance with n jobs, we have $c_{n_i} \le R \cdot \alpha^n$. Applying the claim (1) for $n_i - k_0$ pairs of adjacent completion times implies that

$$\alpha^n = \frac{R \cdot \alpha^n}{R} \ge \frac{c_{n_i}}{c_{k_0}} = \frac{c_{n_i}}{c_{n_i-1}} \cdot \frac{c_{n_i-1}}{c_{n_i-2}} \cdot \ldots \cdot \frac{c_{k_0+1}}{c_{k_0}} \ge \left(\frac{R}{R - \alpha^i}\right)^{n_i - k_0}$$

and, after taking logarithm with base α,

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

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

Let us define

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

The function $f(R, x)$ is a positive and increasing both in R and x for $R > 1$ and $x > 0$. Thus $F(R)$ is increasing as well for $R > 1$. Let R^* be the unique solution of the equation $F(R^*) = 1$ in $(1, +\infty)$.

Theorem 2.2 *For any R-competitive deterministic algorithm for nonpreemptive scheduling on uniformly related machines, it holds that $F(R) \ge 1$. This gives $R \ge R^* > 2.564$.*

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, hence

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

To obtain the last inequality, we bound n_i using Lemma 2.1 as follows: For every $i < \log_\alpha R$ to obtain $n_i \le \lceil R \rceil + \frac{n}{t_i}$. If $i = \log_\alpha R$ is integral, we have $n_i \le 1$. Finally, if $i > \log_\alpha R$ then $n_i = 0$. The inequality follows by summing these bounds.

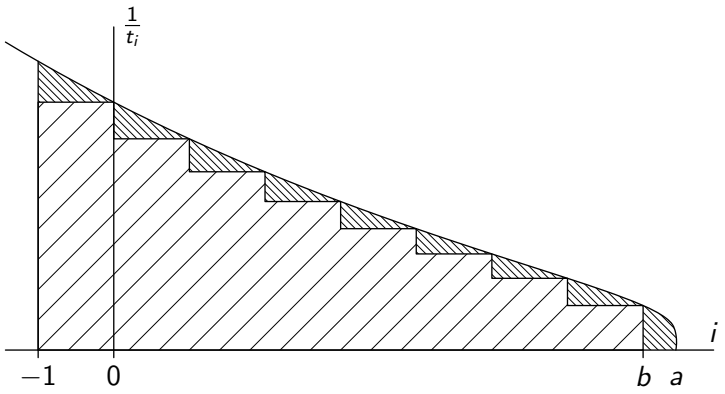

Fig. 1 The labels on the horizontal axis are $a = \log_\alpha R$ and $b = \lceil \log_\alpha R \rceil - 1$. The sparsely hatched region shows the area of the sum in (2). The densely hatched region shows the additional area of the integral in (3).

We can set n arbitrarily large, so that the term $\lceil R \rceil \cdot \lceil \log_\alpha R \rceil + 1$ is negligible. Thus, for any $\varepsilon > 0$, we get:

$$1 - \varepsilon \leq \sum_{i=0}^{\lceil \log_\alpha R \rceil - 1} \frac{1}{t_i} = \sum_{i=0}^{\lceil \log_\alpha R \rceil - 1} \frac{\ln \alpha}{-\ln(1 - \alpha^i R^{-1})} \tag{2}$$

$$\leq \int_{-1}^{\log_\alpha R} \frac{\ln \alpha}{-\ln(1 - \alpha^i R^{-1})}\, di \tag{3}$$

$$= \int_{-\frac{\ln \alpha}{\ln R}}^{1} \frac{\ln R}{-\ln(1 - R^{y-1})}\, dy \tag{4}$$

$$= \int_0^{1 + \frac{\ln \alpha}{\ln R}} f(R, x)\, dx \ . \tag{5}$$

In (3) 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 (4) and $x = 1 - y$ to get (5).

Since $f(R, x)$ is positive and increasing for $x > 0$, it is bounded in the neighborhood of 1 and we have

$$\lim_{\alpha \to 1} \int_0^{1 + \frac{\ln \alpha}{\ln R}} f(R, x)\, dx = \int_0^1 f(R, x)\, dx = F(R) \ .$$

Thus by taking the limits $\alpha \to 1$ and $\varepsilon \to 0$ in (5) we get $1 \leq F(R)$ and the first part of the theorem holds.

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

A natural question is the size of the instance we need in our lower bound. Clearly, to achieve 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 in order 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

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

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

$$\frac{\lceil R'\rceil\lceil\log_\alpha(R')\rceil + 1}{n} \le \varepsilon.$$

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

Numerical calculations show that the constants in the proof can be bounded as follows: 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 be not 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. Subsequent to our work, this was partially achieved in a recent paper [13] using a combination of our analytical approach and the enumerative techniques from [5].

Our techniques cannot be used for lower bounds for the randomized algorithms; the best lower bound in this case remains at 2. Of course, the main challenge in this area is to design new algorithms, perhaps not based on the doubling techniques used so far.

Acknowledgements We are grateful to anonymous reviewers and György Dósa for helpful comments.

References

1. Aspnes, J., Azar, Y., Fiat, A., Plotkin, S., Waarts., O.: On-line load balancing with applications to machine scheduling and virtual circuit routing. J. ACM **44**, 486–504 (1997)
2. Azar, Y.: On-line load balancing. In: A. Fiat, G.J. Woeginger (eds.) Online Algorithms: The State of the Art, pp. 178–195. Springer (1998)
3. Bar-Noy, A., Freund, A., Naor, J.: New algorithms for related machines with temporary jobs. J. Sched. **3**, 259–272 (2000)
4. Berman, P., Charikar, M., Karpinski, M.: On-line load balancing for related machines. In: Proc. 5th Workshop on Algorithms and Data Structures (WADS), *Lecture Notes in Comput. Sci.*, vol. 1272, pp. 116–125. Springer (1997)
5. Berman, P., Charikar, M., Karpinski, M.: On-line load balancing for related machines. J. Algorithms **35**, 108–121 (2000)
6. Cai, S.Y., Yang, Q.F.: Online scheduling on three uniform machines. Discrete Appl. Math. **160**, 291–302 (2011)
7. Cho, Y., Sahni, S.: Bounds for list schedules on uniform processors. SIAM J. Comput. **9**, 91–103 (1980)
8. Ebenlendr, T.: Combinatorial algorithms for online problems: Semi-online scheduling on related machines. Ph.D. thesis, Charles University, Prague (2011)
9. Ebenlendr, T., Jawor, W., Sgall, J.: Preemptive online scheduling: Optimal algorithms for all speeds. Algorithmica **53**, 504–522 (2009)
10. Epstein, L., Noga, J., Seiden, S.S., Sgall, J., Woeginger, G.J.: Randomized on-line scheduling for two uniform machines. J. Sched. **4**, 71–92 (2001)
11. Epstein, L., Sgall, J.: A lower bound for on-line scheduling on uniformly related machines. Oper. Res. Lett. **26**, 17–22 (2000)
12. Han, F., Tan, Z., Yang, Y.: On the optimality of list scheduling for online uniform machines scheduling. Optim. Lett. (2011). To appear
13. Jeż, Ł., Schwartz, J., Sgall, J., Békési, J.: Lower bounds for online makespan minimization on a small number of related machines (2012). To appear in J. Sched.
14. Musitelli, A., Nicoletti, J.M.: Competitive ratio of list scheduling on uniform machines and randomized heuristics. J. Sched. **14**, 89–101 (2011)