A Lower Bound for On-Line Scheduling on Uniformly Related Machines

Leah Epstein Jiří Sgall*

September 23, 1999

lea@math.tau.ac.il, Department of Computer Science, Tel-Aviv University, Israel; sgall@math.cas.cz, Mathematical Inst. AS CR, Žitná 25, CZ-11567 Praha 1, Czech Republic and Dept. of Applied Mathematics, Faculty of Mathematics and Physics, Charles Univ., Praha.

Abstract

We consider the problem of on-line scheduling of jobs arriving one by one on uniformly related machines, with or without preemption. We prove a lower bound of 2, both with and without preemption, for randomized algorithms working for an arbitrary number of machines. For a constant number of machines we give new lower bounds for the preemptive case.

Keywords: On-line scheduling; preemption; uniformly related machines.

1 Introduction

We consider the following scheduling problem. We are given m machines and a sequence of jobs. If a job with processing time p is assigned to a machine of speed s it requires time p/s. In the variation with preemption any job may be divided into several pieces that may be processed on several machines; in addition, the time slots assigned to different pieces must be disjoint. The goal is to minimize the length of the schedule (makespan), i.e., the time when all jobs are finished.

In the on-line problem the jobs arrive in a sequence and we have to assign each job without the knowledge of the future request. The quality of an on-line algorithm is measured by the competitive ratio, which is the worst case ratio of the length of the produced schedule to the optimal (minimal) length.

*Corresponding author. Partially supported by grant A1019901 of GA AV ČR and postdoctoral grant 201/97/P038 of GA ČR.

1.1 Related work

For non-preemptive scheduling on uniformly related machines the first algorithm with a constant competitive ratio was given by Aspnes *et al.* [2]; it is deterministic and its competitive ratio is 8. This was improved by Berman *et al.* [4]; they present 5.8285-competitive deterministic and 4.3111-competitive randomized algorithms. Berman *et al.* [4] also prove lower bounds of 2.4380 for deterministic and 1.8372 for randomized algorithms for non-preemptive scheduling.

For the special case of two related machines the optimal competitive ratio for preemptive scheduling was given independently by Wen and Du [11] and Epstein *et al.* [6] for any combination of speeds. If the ratio of speeds is $s \geq 1$, the optimal competitive ratio is $1 + s/(s^2 + s + 1)$ (this is equal to 4/3 for $s = 1$ and decreases to 1 as $s \to \infty$). Epstein *et al.* [6] also give a 1.53-competitive randomized algorithm for any speeds for non-preemptive scheduling on two related machines and lower bounds for randomized scheduling.

Another special case is an arbitrary number of identical machines (i.e., all the speeds are equal to 1). The preemptive case was settled by Chen *et al.* [5] who gave the tight competitive ratio for any number of machines; the ratio is 4/3 for $m = 2$ and increases to $e/(e-1) \approx 1.582$ as $m \to \infty$, both for deterministic and randomized algorithms. For the nonpreemptive case, for large m the deterministic competitive ratio is known to be between 1.852 and 1.923 [1]. For randomized non-preemptive scheduling for large m no better bounds are known, i.e., the competitive ratio is between $e/(e-1)$ and 1.923. For the numerous results for small m we refer to the survey of on-line scheduling [10]. The variant without preemption is equivalent to load balancing of permanent jobs, for a survey of related results see [3].

For off-line preemptive scheduling the optimal solution was given already by Horvath *et al.* [8] and Gonzales and Sahni [7]. The minimal length of the schedule is the maximum of the following m values: sum of processing times of all jobs divided by the sum of all speeds, and, for $j = 1, \ldots, m-1$, the sum of j largest processing times divided by the sum of j largest speeds. The construction of optimal schedules from [7] also limits the number of preemptions, which allows us to argue that the preemptive schedule is at most 6 times shorter than the non-preemptive one: The algorithm first computes the optimal makespan by the rule above, then processes jobs one by one. Each job is scheduled so that (i) on 2 machines it uses some portion of the remaining processing time and (ii) on all other machines where a non-zero part of the job is scheduled it uses all the remaining processing time. (There are additional rules to guarantee a correct schedule, but those are not important for us.) So, in the non-preemptive schedule we assign the job (i) either to one of the 2 machines if it does more than 1/3 of the work of the job, (ii) or to the fastest machine of the remaining ones where the job is scheduled. For each machine, all the jobs assigned to it by (i) need at most 3 times longer than is the optimal makespan; additionally for each machine there is only one job assigned to it by (ii), and it takes at most 3 times longer than the makespan. The ratio of 6 follows. (The factor of 6 can be tightened; however, even for identical machines the factor is 2, as is shown by the case of $m + 1$ jobs with the same processing times.)

The bound on the factor of optimal non-preemptive and preemptive schedules implies that the above-mentioned non-preemptive on-line algorithms also achieve a constant competitive ratio when considered for the preemptive case, namely 6 times the original competitive ratio. No significantly better preemptive on-line algorithms are known.

1.2 Our results

We prove a lower bound of 2 for preemptive randomized algorithms for scheduling on arbitrary number of uniformly related machines. Since the optimal schedule for the hard instance we use does not use preemption, this lower bound also holds for the non-preemptive randomized algorithms, and improves the bound of 1.8372 given by Berman *et al.* [4].

We also give new lower bounds for any constant number of machines m. It is worth mentioning that while for $m = 2$ the case of identical machines leads to the worst competitive ratio (cf. [11, 6]), for any $m > 3$ we give a larger lower bound than is the optimal competitive ratio for m identical machines.

Our main lemma is a generalization of a lower bound method from [9, 6]. Our hard instance uses machines whose speeds are a geometric sequence and jobs whose processing times are a geometric sequence as well, similarly as in [4].

2 Preliminaries

Let m denote the number of machines, let $s_i > 0$ be the speed of machine M_i, $i = 1, \dots, m$. We assume that $s_1 \geq s_2 \geq \cdots$.

Given a sequence of jobs $\mathcal{J}$, we index the jobs from the end of the sequence, i.e., J_1 is the last job, J_2 the previous one, etc. Let P be the sum of the processing times of all jobs in $\mathcal{J}$. By $\mathcal{J}_i$ we denote the initial segment of $\mathcal{J}$ ending at J_i (i.e., the last $i-1$ jobs are removed; in particular $\mathcal{J}_1 = \mathcal{J}$). The length of an optimal schedule for $\mathcal{J}$ (with or without preemption, depending on which problem we study) is denoted $T_{\mathrm{opt}}(\mathcal{J})$.

For a given randomized algorithm A and a job sequence $\mathcal{J}$, $T_A(\mathcal{J})$ is the length of the schedule it generates on input $\mathcal{J}$; note that it is a random variable. The algorithm A is σ-competitive if for any sequence of jobs $\mathcal{J}$,

$$\mathrm{E}[T_A(\mathcal{J})] \leq \sigma \cdot T_{\mathrm{opt}}(\mathcal{J}),$$

where $\mathrm{E}[T_A(\mathcal{J})]$ denotes the expected length of the schedule generated by A.

We also consider a variant with infinitely many machines and jobs. In this case we assume that the sum of all speeds, $\sum_{i=1}^{\infty} s_i$, is finite, and the sum of all processing times P is finite as well. In the variant with an infinite number of machines we allow the sequence of jobs to be only backwards infinite, i.e., J_i, $i = 1, 2, \ldots$, enumerates all the jobs starting with the last one. The infinite variant may be at first somewhat strange, but it provides a sound intuition, and simplifies the analysis significantly. Alternatively, we can interpret the infinite variant as follows: instead of an infinite instance of the problem consider only a finite number of largest jobs and fastest machines; as this number increases, all the bounds converge to the bounds given for this infinite instance.

We first prove a general lemma which applies to any sequence of jobs; it works also for the infinite variant if m is replaced by ∞ in the sums.

Lemma 2.1 *For any randomized σ-competitive on-line algorithm A for scheduling on m machines, with or without preemption, we have*

$$P \leq \sum_{i=1}^{m} s_i \cdot \mathrm{E}[T_A(\mathcal{J}_i)] \leq \sigma \sum_{i=1}^{m} s_i \cdot T_{\mathrm{opt}}(\mathcal{J}_i).$$

Proof. Fix a sequence of random bits used by the algorithm A. Let T_i be the last time when at least i machines are running.

First, we claim that

$$P \leq \sum s_i T_i. \tag{1}$$

During the times in the time interval from T_{i+1} to T_i, at most i machines are busy. Thus the total work done during this interval is at most $(T_i - T_{i+1})(s_1 + \cdots + s_i)$, as the maximum is obtained if the busy machines are the fastest ones. Summing over all time intervals (with a similar consideration for the interval from 0 to $\inf T_i$) yields that the total running time of jobs processed is $\sum_{i=1}^{m} s_i T_i$. The schedule must process all jobs, with total running time P, and therefore (1) holds.

Since the algorithm is on-line, the schedule for $\mathcal{J}_i$ is obtained from the schedule for $\mathcal{J}$ by removing the last $i-1$ jobs. At time T_i there are at least i jobs running, thus even after removing $i-1$ jobs from the schedule at least one job running at time T_i remains and $T_i \leq T_A(\mathcal{J}_i)$. Combining this bound with (1) and averaging over the random bits of the algorithm we obtain

$$P \leq \sum_{i=1}^{m} s_i \cdot \mathrm{E}[T_A(\mathcal{J}_i)].$$

The assumption that the algorithm A is σ-competitive implies that $\mathrm{E}[T_A(\mathcal{J}_i)] \leq \sigma \cdot T_{\mathrm{opt}}(\mathcal{J}_i)$, and the lemma follows. □

3 Unbounded number of machines

Theorem 3.1 *For any randomized on-line algorithm for scheduling on arbitrary number of uniformly related machines, the competitive ratio is at least* 2*. This lower bound holds both with and without preemption.*

Proof. Let $x < 1$ be given. We consider infinitely many machines with speeds $s_i = x^i$ and an infinite sequence of jobs with processing times $p_j = x^j$. We have

$$P = \sum_{j=1}^{\infty} p_j = \frac{x}{1-x}.$$

Now consider the sequence of jobs $\mathcal{J}_i$, i.e., without the last $i-1$ jobs. In the optimal schedule, job j is scheduled on machine $j-i+1$, and the load of each machine is $p_j/s_{j-i+1} = x^{i-1}$. Thus $T_{\text{opt}}(\mathcal{J}_i) = x^{i-1}$, for all i, both with and without preemption. By Lemma 2.1,

$$\frac{x}{1-x} = P \leq \sigma \cdot \sum_{i=1}^{\infty} s_i \cdot T_{\text{opt}}(\mathcal{J}_i) = \sigma \cdot \sum_{i=1}^{\infty} x^{2i-1} = \sigma \cdot \frac{x}{1-x^2}.$$

Therefore

$$\sigma \geq \frac{x}{1-x} \cdot \frac{1-x^2}{x} = 1+x.$$

Since x can be arbitrarily close to 1, the theorem follows. □

4 Constant number of machines

We first get a simple bound and then improve it. Both bounds build on the intuition from the infinite case. The simple one just takes the m largest jobs and fastest machines. The better bound adjusts the speeds and jobs so that both the sum of all speeds and the sum of all processing times are the same as in the infinite case.

For the simple bound, consider m jobs with processing times $t_j = x^j$ and m machines with speeds $s_i = x^i$ for $0 < x < 1$. We have $P = x(1-x^m)/(1-x)$, $T_{\text{opt}}(\mathcal{J}_i) = x^{i-1}$ and

$$\sum_{i=1}^{m} s_i T_{\text{opt}}(\mathcal{J}_i) = \sum_{i=1}^{m} x^{2i-1} = \frac{x(1-x^{2m})}{(1-x^2)}.$$

Thus the competitive ratio is at least $(1+x)/(1+x^m)$. For m going to infinity and $x < 1$ a constant the limit is $1+x$.

For a better bound for a constant m we slightly modify the sequence of jobs and speeds. Note that the new sequence for $x = 1-1/m$ gives the hard instance for identical machines.

Let $0 < x \leq 1-1/m$ be a parameter. We have m large jobs with processing times $p_j = x^j$ and then some number of small jobs with total processing time $x^{m+1}/(1-x)$. Thus the total processing time of all jobs is $P = x/(1-x)$. The speeds are chosen as follows: for some k and z (to be determined later) $s_i = x^i$ for $i \leq k$ and $s_i = z$ for $i > k$. The values of k and z are chosen so that the sum of all speeds satisfies $\sum_{i=1}^{m} s_i = x/(1-x)$, and $x^{k+1} \leq z < x^k$. For each $x \leq 1-1/m$ there exists a unique pair of k and z satisfying the condition, and it is given by

$$\begin{aligned} k &= m - \left\lfloor \frac{1}{1-x} \right\rfloor, \qquad (2) \\ z &= \frac{x^{k+1}}{(m-k)(1-x)}. \end{aligned}$$

It is easy to verify that $T_{\text{opt}}(\mathcal{J}_i) = x^{i-1}$, both with and without preemption.

Lemma 2.1 shows that

$$\frac{1}{\sigma} \leq \frac{\sum_{i=1}^m s_i \cdot T_{\mathrm{opt}}(\mathcal{J}_i)}{P} = \frac{\sum_{i=1}^k x^{2i-1} + \sum_{i=k+1}^m z x^{i-1}}{\frac{x}{1-x}}$$

$$= \frac{\frac{x-x^{2k+1}}{1-x^2} + \frac{x^{k+1}(x^k - x^m)}{(m-k)(1-x)^2}}{\frac{x}{1-x}} = \frac{1-x^{2k}}{1+x} + \frac{x^{2k} - x^{m+k}}{(m-k)(1-x)}. \tag{3}$$

We have optimized this expression numerically using Mathematica. Below are a few optimal values of k, x and (lower bounds on) σ.

m	k	x	lower bound on σ
2	1	0.500000	1.33333
3	1	0.548584	1.46106
4	1	0.641564	1.55033
5	2	0.669949	1.61170
6	2	0.716886	1.65474
7	2	0.749478	1.69093
8	3	0.764008	1.71774
9	3	0.785953	1.74056
10	3	0.799998	1.76027
50	17	0.941736	1.93240
100	30	0.967172	1.96234

The following theorem summarizes our results for a constant number of machines.

Theorem 4.1 *For any randomized on-line algorithm for scheduling on m uniformly related machines, the competitive ratio σ satisfies the inequality (3) for an arbitrary x, $0 < x \leq 1 - 1/m$, and k defined by (2). In particular, we obtain the lower bounds given in the table above. These lower bounds hold both with and without preemption.*

Note that for $m = 2$ the case of identical machines leads to the worst case competitive ratio, while for any $m > 3$ identical machines are not the hardest case: For $m = 3, 4$ the ratio for the identical machines is 1.421 and 1.463 [5], respectively; for $m > 4$ our preemptive lower bounds are larger than $e/(e-1)$, which is the upper bound for an arbitrary number of identical machines.

Conclusions and open problems

Our results leave many problems open. Mainly, we have no (good) algorithms for on-line preemptive scheduling on related machines.

The special cases of two related machines and m identical machines both suggest that the preemptive algorithms are generally easier to design than non-preemptive ones and achieve a better competitive ratio. However, here we even do not achieve the same upper bounds as in the non-preemptive case. (Recall that the best algorithms for preemptive scheduling we have are the non-preemptive ones, using the fact that the ratio of the optimal preemptive solution and the non-preemptive one is bounded by a constant.) This is fairly unsatisfactory. We conjecture that the preemptive competitive ratio should be at most the non-preemptive randomized one. It is even plausible that there exists a 2-competitive preemptive algorithm.

One difficulty is that the known (non-preemptive) algorithms rely on the fact that the optimal solution schedules each job on a single machine: The doubling algorithm of [2] guesses the optimal value, then schedules each job on the slowest machine where it would finish before twice the guess; the underlying ideas in [4] are similar. In this framework it is hard to take an advantage of the fact that we are allowed to preempt a job. On the other hand, those algorithms only guess the optimal value; good preemptive algorithms should take advantage of the fact that the preemptive optimum can be computed exactly.

As far as lower bounds are concerned, we believe that our analysis is tight for the given set of speeds for preemptive scheduling, at least for the infinite variant. It is not clear at all what happens if the speeds are not a geometric sequence—even our main lemma could possibly yield better lower bounds.

References

[1] S. Albers. Better bounds for online scheduling. In *Proc. of the 29th Ann. ACM Symp. on Theory of Computing*, pages 130–139. ACM, 1997.

[2] J. Aspnes, Y. Azar, A. Fiat, S. Plotkin, and O. Waarts. On-line load balancing with applications to machine scheduling and virtual circuit routing. *J. Assoc. Comput. Mach.*, 44(3):486–504, 1997.

[3] Y. Azar. On-line load balancing. In A. Fiat and G. J. Woeginger, editors, *Online Algorithms: The State of the Art*, volume 1442 of *Lecture Notes in Comput. Sci.*, pages 178–195. Springer-Verlag, 1998.

[4] P. Berman, M. Charikar, and M. Karpinski. On-line load balancing for related machines. In *Proc. of the 5th Workshop on Algorithms and Data Structures, Lecture Notes in Comput. Sci. 1272*, pages 116–125. Springer-Verlag, 1997.

[5] B. Chen, A. van Vliet, and G. J. Woeginger. An optimal algorithm for preemptive on-line scheduling. *Oper. Res. Lett.*, 18:127–131, 1995.

[6] L. Epstein, J. Noga, S. S. Seiden, J. Sgall, and G. J. Woeginger. Randomized online scheduling for two related machines. In *Proc. of the 10th Ann. ACM-SIAM Symp. on Discrete Algorithms*, pages 317–326. ACM-SIAM, 1999.

[7] T. F. Gonzales and S. Sahni. Preemptive scheduling of uniform processor systems. *J. Assoc. Comput. Mach.*, 25:92–101, 1978.

[8] E. Horwath, E. C. Lam, and R. Sethi. A level algorithm for preemptive scheduling. *J. Assoc. Comput. Mach.*, 24:32–43, 1977.

[9] J. Sgall. A lower bound for randomized on-line multiprocessor scheduling. *Inf. Process. Lett.*, 63(1):51–55, 1997.

[10] J. Sgall. On-line scheduling. In A. Fiat and G. J. Woeginger, editors, *Online Algorithms: The State of the Art*, volume 1442 of *Lecture Notes in Comput. Sci.*, pages 196–231. Springer-Verlag, 1998.

[11] J. Wen and D. Du. Preemptive on-line scheduling for two uniform processors. *Oper. Res. Lett.*, 23:113–116, 1998.