Information Processing Letters 76 (2000) 149–153

Information Processing Letters

www.elsevier.com/locate/ipl

# A note on on-line scheduling with precedence constraints on identical machines [☆]

Leah Epstein

*Department of Computer Science, Tel-Aviv University, Ramat-Aviv, Israel*

Received 26 August 1998; received in revised form 14 August 2000
Communicated by S. Zaks

**Abstract**

We consider the on-line scheduling problem of jobs with precedence constraints on $m$ parallel identical machines. Each job has a time processing requirement, and may depend on other jobs (has to be processed after them). A job arrives only after its predecessors have been completed. The cost of an algorithm is the time at which the last job is completed. We show lower bounds on the competitive ratio of on-line algorithms for this problem in several versions. We prove a lower bound of $2-1/m$ on the competitive ratio of any deterministic algorithm (with or without preemption) and a lower bound of $2-2/(m+1)$ on the competitive ratio of any randomized algorithm (with or without preemption). The lower bounds for the cases that preemption is allowed require arbitrarily long sequences. If we use only sequences of length $O(m^2)$, we can show a lower bound of $2-2/(m+1)$ on the competitive ratio of deterministic algorithms with preemption, and a lower bound of $2-O(1/m)$ on the competitive ratio of any randomized algorithm with preemption. All the lower bounds hold even for sequences of unit jobs only. The best algorithm that is known for this problem is the well-known List Scheduling algorithm of Graham. The algorithm is deterministic and does not use preemption. The competitive ratio of this algorithm is $2-1/m$. Our randomized lower bounds are very close to this bound (a difference of $O(1/m)$) and our deterministic lower bounds match it. © 2000 Elsevier Science B.V. All rights reserved.

*Keywords:* Algorithms; Competitive ratio; Lower bounds

## 1. Introduction

We consider the problem of scheduling a sequence of $n$ jobs on $m$ parallel identical machines. There are precedence constraints between the jobs, which can be given by a directed acyclic graph on the jobs. In this graph each directed edge from job $j_1$ to job $j_2$ indicates that $j_1$ has to be scheduled before $j_2$. We consider an on-line environment in which a job becomes known only upon completion of all its predecessors (several jobs may become known simultaneously). Each job $j$ has a certain time requirement $w_j$ (which is known when the job arrives). An on-line algorithm does not have to assign a job immediately when it becomes known. The cost of an algorithm is the makespan, which is the time at which the last job is completed. This model is realistic since often the running times of specific jobs are known in advance, but it is unknown whether after these jobs, there will be a need to perform jobs that depend on some previous jobs.

[☆] A preliminary version of this work appeared in the Proceedings of the 1st Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX'98) pp. 89–98.

*E-mail address:* lea@math.tau.ac.il (L. Epstein).

We compare the performance of on-line algorithms and the optimal off-line algorithm that knows the sequence of jobs and the dependence graph in advance. We use the competitive ratio to measure the performance of the algorithm. Denote the cost of the on-line algorithm by $C_{\mathrm{on}}$ and the cost of the optimal off-line algorithm by $C_{\mathrm{opt}}$. The competitive ratio of the on-line algorithm is $r$ if for any input sequence of jobs: $C_{\mathrm{on}} \leqslant r \cdot C_{\mathrm{opt}}$. For randomized algorithms the competitive ratio is $r$ if for any input sequence of jobs: $E(C_{\mathrm{on}}) \leqslant r \cdot C_{\mathrm{opt}}$.

We consider several models. A job $j$ which has processing time requirement $w_j$ has to be processed on one machine for $w_j$ time units. In the preemptive model, a running job can be stopped and resumed later on the same or different machine. Thus $j$ still has to be processed for a total of $w_j$ time units, but not necessarily continuously, nor on one machine (it cannot be processed on more than one machine at the same time). It is also possible to consider special cases such as sequences that consist only of unit jobs, and sequences of bounded length. A summary of results for many variants of on-line scheduling problems appears in [15].

The first to introduce the on-line scheduling problem was Graham [9,10]. He also introduced the algorithm List Scheduling. This algorithm schedules an available job (a job which was not scheduled yet but already arrived) if such a job exists, whenever a machine becomes idle. List Scheduling has the competitive ratio of $2 - 1/m$, which is valid even if durations are not known upon arrival. In this paper we show that for our problem, the algorithm is optimal in the deterministic case, and that it is almost optimal among randomized algorithms.

Our deterministic lower bounds build on the paper of Shmoys et al. [16]. They consider the problem of scheduling a sequence of independent tasks on-line. The duration of a job is unknown until it is completed, but there are no precedence constraints between the jobs. They show a lower bound of $2 - 1/m$ on the competitive ratio of any deterministic algorithm without preemption. In this paper we adapt this lower bound to a lower bound of $2 - 1/m$ for our problem. The problem of preemptive scheduling with precedence constraints is at least as hard as preemptive scheduling with unknown durations. This is true since a job of unknown duration can be seen as a chain (directed path in the dependence graph) of small jobs. The lower bound proof of [16] for deterministic preemptive scheduling is based on the assumption that the adversary can stop a job at any moment. However the reduction assumes a lower bound of some small $\varepsilon > 0$ on the time requirement of every job. Thus the lower bound for the unknown durations model does not give a straightforward lower bound for the model of known durations and precedence constraints. We give an alternative lower bound proof, which gives the same bound of $2 - 1/m$, its proof is also built on some ideas of the non-preemptive lower bound of [16]. Shmoys et al. also give a lower bound of $2 - \mathrm{O}(1/\sqrt{m})$ on the competitive ratio of randomized algorithms without preemption. The last lower bound can be also adapted to our problem (even with preemption), but we show a much stronger lower bound for randomized algorithms.

### 1.1. *Our results*

All the results apply even if the sequences may consist of unit jobs only. Also, the number of jobs, the structure and makespan of the optimal off-line assignment are known in advance in all lower bounds.

We prove the following lower bounds for *deterministic algorithms*: A lower bound of $2 - 1/m$ on the competitive ratio of any deterministic on-line algorithm without preemption (even if the length of the sequence is limited to $\mathrm{O}(m^2)$), a lower bound of $2 - 1/m$ on the competitive ratio of any deterministic on-line algorithm that allows preemption, and a lower bound of $2 - 2/(m + 1)$ on the competitive ratio of any deterministic on-line algorithm that allows preemption (even if the length of the sequence is limited to $\mathrm{O}(m^2)$).

We prove the following lower bounds for *randomized algorithms*: A lower bound of $2 - 2/(m + 1)$ on the competitive ratio of any randomized on-line algorithm without preemption (even if the length of the sequence is limited to $\mathrm{O}(m^2)$), a lower bound of $2 - 2/(m + 1)$ on the competitive ratio of any randomized on-line algorithm that allows preemption, and a lower bound of $2 - \mathrm{O}(1/m)$ on the competitive ratio of any randomized on-line algorithm that allows preemption (even if the length of the sequence is limited to $\mathrm{O}(m^2)$).

The similar results for all the models show that for this problem, neither randomization nor preemption can significantly help in reducing the competitive ratio. Any algorithm would have the competitive ratio of $2 - O(1/m)$ which is very close to the competitive ratio of List Scheduling.

## 2. Deterministic algorithms

In this section we show a lower bound of $2-1/m$ on the competitive ratio of deterministic algorithms with preemption.

**Theorem 2.1.** *The competitive ratio of any deterministic algorithm with or without preemption is at least* $2 - 1/m$. *This is true even if all jobs are unit jobs. If the sequence is limited to* $O(m^2)$ *jobs, the competitive ratio of any deterministic algorithm* (*with or without preemption*) *is at least* $2 - 2/(m+1)$.

**Proof.** We start with the non-preemptive lower bound. We use the following sequence (all jobs are unit jobs). First $m(m-1)+1$ jobs arrive. Consider the on-line assignment. Since the total time to schedule $m(m-1)$ jobs is at least $m-1$, there is at least one job $j$ assigned at time $m-1$ or later, and this job finishes at time $m$ or later.

After that, a sequence $J$ of $m-1$ jobs $j_1, \ldots, j_{m-1}$ arrives. In this sequence the first job $j_1$ depends on $j$, and each job $j_i$ depends on the previous one $j_{i-1}$, $2 \leqslant i \leqslant m-1$. It takes an additional $m-1$ units of time to schedule them and thus $C_{\mathrm{on}} \geqslant m + (m-1) = 2m-1$.

The optimal off-line algorithm would schedule $j$ at time 0, and each $j_i$ at time $i$ and thus can finish all $m^2$ jobs in $m$ time units. Thus $C_{\mathrm{opt}} = m$ and the competitive ratio is $2 - 1/m$.

Allowing preemptions does not change $C_{\mathrm{opt}}$ but might allow the on-line algorithm to complete the first batch of jobs in time $m - 1 + 1/m$, leading to a lower bound of $2 - 2/m + 1/m^2$. We can obtain a better bound by having $(m^k+1)(m-1)+1$ jobs in the first batch and $m^k$ jobs in $J$, for some integer $k$. Using the same reasoning as before, we obtain a lower bound of

$$2 - \frac{m^{k-1}+1}{m^k+1} = 2 - \frac{1}{m} - \varepsilon_k,$$

where $\varepsilon_k \to 0$ for $k \to \infty$. By putting $k=1$ we get a bound of $2 - 2/(m+1)$ using a sequence of $O(m^2)$ jobs. $\square$

## 3. Randomized algorithms

In all the proofs in this section, which are lower bounds on the competitive ratio of randomized algorithms, we use an adaptation of Yao's theorem for on-line algorithms. It states that if there exists a probability distribution on the input sequences for a given problem such that $E(C_{\mathrm{on}}/C_{\mathrm{opt}}) \geqslant c$ for all deterministic on-line algorithms, then $c$ is a lower bound on the competitive ratio of all randomized algorithms for the problem. We will use only sequences for which $C_{\mathrm{opt}}$ is constant and thus in our case

$$E(C_{\mathrm{on}}/C_{\mathrm{opt}}) = E(C_{\mathrm{on}})/C_{\mathrm{opt}}.$$

We begin with a lower bound without preemption. Note that in this section the lower bound without preemption uses a totally different sequence than the lower bound with preemption. As before, all lower bound sequences consist of unit jobs only.

**Theorem 3.1.** *The competitive ratio of any randomized algorithm without preemption is at least* $2 - 2/(m+1)$. *This is true even if all jobs are unit jobs.*

**Proof.** First $m-1$ phases of $m+1$ jobs arrive. In each phase, all jobs depend on $m$ jobs of the previous phase. (For each phase, the subset of $m$ jobs from the previous phase is chosen among all

$$\binom{m+1}{m} = m+1$$

possible subsets of $m$ jobs with equal probability.) After all $m^2-1$ jobs have arrived, another job that depends on m jobs of the last phase arrives (here too the $m$ jobs are chosen among the $m+1$ possible subsets with equal probability).

For each phase $j$, $0 \leqslant j \leqslant m-2$, the optimal off-line algorithm schedules the $m$ jobs that the next phase depends on at time $j$. All other jobs are scheduled at time $m-1$ and thus $C_{\mathrm{opt}} = m$.

Consider the arrival of a phase at time $i$. Since the next phase can arrive only after the $m$ “important” jobs in the current phase complete, there are two possibilities. If the algorithm schedules $m$ of the current jobs immediately and these jobs happen to be the important ones, the next phase will arrive at time $i+1$. Otherwise, the next phase will not arrive before time $i+2$. The first scenario occurs with probability at most $1/(m+1)$ and thus the expected contribution of the phase to $C_{\text{on}}$ is at least $1/(m+1)+2m/(m+1)$. Thus, counting the $m-1$ phases and the last job, $E(C_{\text{on}}) \geqslant 2m^2/(m+1)$ and $E(C_{\text{on}})/C_{\text{opt}} \geqslant 2-2/(m+1)$. □

Note that the construction in the previous proof can serve as an alternative means to prove the lower bound of $2-1/m$ for deterministic algorithms without preemption.

**Theorem 3.2.** *The competitive ratio of any randomized algorithm with preemption is at least* $2-2/(m+1)$. *This is true even if all jobs are unit jobs.*

**Proof.** For an integer $k$, $N=(m^k+1)(m-1)+1$ jobs arrive. Denote $L=N/m=m^k+1-m^{k-1}$. An additional $m^k$ jobs: $j_1,\dots,j_{m^k}$ arrive so that $j_1$ depends on a subset $J$ of $m$ of the $N$ jobs, which is chosen uniformly at random among all $\binom{N}{m}$ possible subsets, and for $1 \leqslant i \leqslant m^k-1$, $j_{i+1}$ depends on $j_i$.

Even with preemption, since each job requires one processing unit, at most $im$ jobs can finish by time $i$. Let $b_1,\dots,b_N$ be the set of the first $N$ jobs sorted according to their finishing time ($b_1$ finishes first). For $1 \leqslant i \leqslant L$, let $J_i$ be the set $b_{im-m+1},\dots,b_{im}$. Each job in a set $J_i$ cannot finish before time $i-1$. Let $I$ be the set of indices $i$ such that there is at least one job of $J$ in $J_i$. Let $i_1$ be the maximum index in $I$. We define $p_i$ to be the probability that $i=i_1$ for each $1 \leqslant i \leqslant L$. For $0 \leqslant i \leqslant L$, let $q_i$ be the probability that all $m$ jobs of $J$ are chosen among the jobs in the sets $J_1,\dots,J_i$ (we define $q_0=0$). Then

$$q_i = \binom{mi}{m} \Big/ \binom{N}{m} \quad \text{and} \quad p_i = q_i - q_{i-1}.$$

Let us calculate $E(C_{\text{on}})$. For a fixed value of $i_1$ the on-line cost is at least $i_1-1+m^k$.

$$\begin{aligned}
E(C_{\text{on}}) &\geqslant \sum_{i=1}^{L} p_i(i-1+m^k) \\
&= (m^k-1)\sum_{i=1}^{L} p_i + \sum_{i=1}^{L} i(q_i-q_{i-1}) \\
&= m^k-1+\sum_{i=1}^{L} iq_i - \sum_{i=1}^{L} i(q_{i-1}) \\
&= m^k-1+Lq_L+\sum_{i=1}^{L-1} iq_i - \sum_{i=0}^{L-1}(i+1)q_i \\
&= m^k-1+L-\sum_{i=1}^{L-1} q_i - q_0 \\
&= m^k-1+L-\sum_{i=1}^{L-1} q_i.
\end{aligned}$$

(Since $\sum_{i=1}^{L} p_i = 1$, $q_L=1$ and $q_0=0$.) Let us bound the values $q_i$:

$$q_i = \frac{\binom{mi}{m}}{\binom{N}{m}} = \frac{(mi)!(N-m)!}{(N)!(mi-m)!} \leqslant \left(\frac{i}{L}\right)^m.$$

We use the following inequality of Bernoulli:

$$L^{m+1} \geqslant (L-1)^{m+1} + (m+1)(L-1)^m.$$

By induction we can get

$$L^{m+1} \geqslant \sum_{i=1}^{L-1} (m+1)i^m$$

hence

$$\sum_{i=1}^{L-1} q_i \leqslant \frac{L}{m+1}.$$

Thus

$$\begin{aligned}
E(C_{\text{on}}) &\geqslant m^k-1+L-\frac{L}{m+1} \\
&= m^k-1+\left(1-\frac{1}{m+1}\right)(m^k-m^{k-1}+1) \\
&= \frac{2m^{k+1}-1}{m+1}.
\end{aligned}$$

The optimal off-line algorithm would assign all jobs in $J$ at time 0, and get $C_{\text{opt}}=m^k+1$. Thus the competitive ratio is at least $(2m^{k+1}-1)/((m+1)(m^k+1))$. Since $(m^k-(2m)^{-1})/(m^k+1) \to 1$ when $k \to \infty$, the competitive ratio is at least $2m/(m+1) = 2-2/(m+1)$. □

If we wish to restrict ourselves to sequences of $O(m^2)$ jobs, we can set $k = 1$. The expression bounding $q_i$ then becomes $(i/m)^m < e^{i-m}$. Thus,

$$\sum_{i=1}^{m-1} q_i \leqslant \frac{e(e^{m-1}-1)}{e^m(e-1)} \leqslant \frac{1}{e-1}$$

and

$$E(C_{\mathrm{on}}) \geqslant 2m - 1 - \frac{1}{e-1}.$$

The competitive ratio is thus $2 - O(1/m)$.

## Acknowledgements

I would like to thank Yossi Azar and Jiří Sgall for helpful discussions.

## References

[1] Y. Azar, L. Epstein, On-line scheduling with precedence constraints, Manuscript, Tel-Aviv University, 1999.
[2] B. Chen, A. Vestjens, Scheduling on identical machines: How good is lpt in an on-line setting?, Oper. Res. Lett. 21 (4) (1997) 165–169.
[3] F.A. Chudak, D.B. Shmoys, Approximation algorithms for precedence constrained scheduling problems on parallel machines that run at different speeds, in: Proc. 8th Ann. ACM-SIAM Symp. on Discrete Algorithms, 1997, pp. 581–590.
[4] E. Davis, J.M. Jaffe, Algorithms for scheduling tasks on unrelated processors, J. ACM 28 (4) (1981) 721–736.
[5] A. Feldmann, M.-Y. Kao, J. Sgall, S.-H. Teng, Optimal online scheduling of parallel jobs with dependencies, in: Proc. 25th Ann. ACM Symp. on Theory of Computing, pp. 642–651.
[6] A. Feldmann, B. Maggs, J. Sgall, D.D. Sleator, A. Tomkins, Competitive analysis of call admission algorithms that allow delay, Technical Report CMU-CS-95-102, Carnegie-Mellon University, Pittsburgh, PA, 1995.
[7] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-completeness, W.H. Freeman, New York, 1979.
[8] T. Gonzalez, D.B. Johnson, A new algorithm for preemptive scheduling of trees, J. ACM 27 (1980) 287–312.
[9] R.L. Graham, Bounds for certain multiprocessor anomalies, Bell System Techn. J. 45 (1966) 1563–1581.
[10] R.L. Graham, Bounds on multiprocessing timing anomalies, SIAM J. Appl. Math. 17 (1969) 263–269.
[11] K.S. Hong, J.Y.-T. Leung, On-line scheduling of real-time tasks, IEEE Trans. Comput. 41 (10) (1992) 1326–1331.
[12] J.M. Jaffe, Efficient scheduling of tasks without full use of processor resources, Comput. Sci. 12 (1980) 1–17.
[13] J.W.S. Liu, C.L. Liu, Bounds on scheduling algorithms for heterogeneous computing systems, in: Inform. Process. 74, North-Holland, Amsterdam, 1974, pp. 349–353.
[14] S. Sahni, Y. Cho, Nearly on line scheduling of a uniform processor system with release times, SIAM J. Comput. 8 (2) (1979) 275–285.
[15] J. Sgall, On-line scheduling, in: A. Fiat, G.J. Woeginger (Eds.), Online Algorithms: The State of the Art, Lecture Notes in Comput. Sci., Vol. 1442, Springer, Berlin, 1998, pp. 196–231.
[16] D.B. Shmoys, J. Wein, D.P. Williamson, Scheduling parallel machines on line, SIAM J. Comput. 24 (1995) 1313–1331.
[17] A.P.A. Vestjens, Scheduling uniform machines on-line requires nondecreasing speed ratios, Math. Programming 82 (1–2) (1998) 225–234.
[18] A.P.A. Vestjens, On-line machine scheduling, Ph.D. Thesis, Eindhoven University of Technology, The Netherlands, 1997.