On-line Scheduling with Precedence Constraints *

Yossi Azar†
Tel-Aviv Univ.

Leah Epstein‡
Tel-Aviv Univ.

Abstract

We consider the on-line problem of scheduling jobs with precedence constraints on m machines. We concentrate in two models, the model of uniformly related machines and the model of restricted assignment. For the related machines model, we show a lower bound of $\Omega(\sqrt{m})$ for the competitive ratio of deterministic and randomized on-line algorithms, with or without preemptions even for known running times. This matches the deterministic upper bound of $O(\sqrt{m})$ given by Jaffe. The lower bound should be contrasted with the known bounds for jobs without precedence constraints. Specifically, without precedence constraints, if we allow preemptions then the competitive ratio becomes $\Theta(\log m)$, and if the running times of the jobs are known then there are $O(1)$ competitive (preemptive and non-preemptive) algorithms.

We also consider the restricted assignment model. For the model with consistent precedence constraints, we give a (randomized) lower bound of $\Omega(\log m)$ with or without preemptions. We show that a (deterministic, non-preemptive) greedy algorithm is *optimal* up to a constant factor for this model i.e. $O(\log m)$ competitive. However, for general precedence constraints, we show a lower bound of m which is easily matched by a greedy algorithm.

*A Preliminary version of this paper appears in the proceedings of the 7th Biennial Scandinavian Workshop on Algorithm Theory, SWAT'2000 pp. 164-174.

†Dept. of Computer Science, Tel-Aviv University. E-Mail: azar@tau.ac.il. Research supported in part by the Israel Science Foundation and by the United States-Israel Binational Science Foundation (BSF).

‡Dept. of Computer Science, Tel-Aviv University. E-Mail: lea@tau.ac.il

1 Introduction

We consider the on-line problem of scheduling a sequence of jobs with precedence constraints on m parallel machines. A job can be scheduled after all its predecessors are completed. In the simplest model, the identical machines model, each job j has a running time w_j, and has to be scheduled on a machine for this period of time.

In the related machines model each machine i has a speed v_i. Each job may be processed on any machine. If job j with a running time w_j is processed on machine i then the job is completed in w_j/v_i units of time. In the restricted assignment model all machines have identical speed, but each job may be assigned only to a subset of the machines. For a job j, we denote by $M(j) \subseteq \{1, \ldots, m\}$ $(M(j) \neq \emptyset)$ the subset of machines on which it may be scheduled and by w_j its running time on a machine in $M(j)$. The unrelated machines model is a generalization of all previous models. In this model, each job j has a vector of m components, where each component i gives its running time on machine i.

We may or may not allow preemptions. If no preemptions are allowed, once a job is scheduled on a machine, it must be processed on this machine continuously until it is completed. Otherwise, if we allow preemptions, a job may be stopped, and resumed later on some (maybe different) machine.

The precedence constraints between jobs can be viewed as a directed acyclic graph G. The vertices of G are the jobs. An edge (j_1, j_2) occurs when j_1 is a predecessor of j_2, i.e. j_2 may start its process only after j_1 is completed. For restricted assignment the precedence constraints are called consistent if for every edge (j_1, j_2) we have $M(j_2) \subseteq M(j_1)$. The motivation for consistent precedence constraints comes from the fact that if a job j_1 requires some expertise which are known only to some machines and j_1 is a predecessor of another job j_2, then j_2 should require at least the same expertise and hence can be processed only on subset of machines that j_1 can be processed on.

We discuss an on-line environment in which a job becomes known as soon as all its predecessors are completed (there are no release times). The goal is to minimize the makespan which is the time that the last job is completed. We consider two variations of the on-line model. In the known running times case (also called clairvoyant case), the running time of a job is known upon its arrival, and in the unknown running times case (also called non-clairvoyant case), the running time of a job becomes known only when it is completed. Clearly an algorithm for the unknown running times case is also an algorithm for the known running times case with the same performance. Hence, lowers bounds for known running times are lower bounds also for unknown running times. For a survey on on-line scheduling we refer the reader to [12].

We measure the algorithms in terms of the competitive ratio. We compare the cost (makespan) of the on-line algorithm (denoted by C_{on}) to the cost of the optimal off-line algorithm that knows the sequence in advance (denoted by C_{opt}). The off-line algorithm knows all jobs and their properties (running times, precedence constraints and assignment restrictions) in advance. Note that the on-line algorithm is familiar with all properties of a job as soon as the job arrives (except for the running time, in the case of unknown running times), but a job arrives only after all its predecessors are completed. A deterministic algorithm is r competitive if $C_{on} \leq rC_{opt}$. The competitive ratio of an algorithm is the infimum r such that the algorithm is r competitive. If the algorithm is randomized, we use the expectation of the on-line cost instead of the cost, and an algorithm is r competitive if $E(C_{on}) \leq rC_{opt}$.

The model with precedence constraints generalizes the model without precedence constraints, i.e. the model where all jobs are given at time 0. The model without precedence constraints is also called batch-style scheduling. The batch-style scheduling is important since the general model of jobs arriving over time with release times can be reduced to the batch-style scheduling with a loss of a factor of 2 in the approximation ratio or in the competitive ratio (see [13]). We note that the batch-style scheduling is interesting for on-line algorithms only when the running times of the jobs are unknown. If the running times are known then the batch-style scheduling becomes an off-line problem. In contrast, the model with precedence constraints is also interesting for the known running times case since not all jobs are given in advance.

The identical machines model is an ancient on-line problem. Graham [7, 8] showed a greedy (non-preemptive) algorithm that achieves a competitive ratio of $2 - 1/m$ for scheduling jobs with precedence constraints even when the running times are unknown. Epstein [5] showed that this is optimal for scheduling jobs with precedence constraints even if the running times are known and preemptions are allowed. The best competitive ratios for the three other classical machine models, related machines, restricted assignment and unrelated machines, were not completely characterized.

Our results. In this paper we consider the three classical machine models: related machines, restricted assignment and unrelated machines. For related machines we give a deterministic and randomized lower bound of $\Omega(\sqrt{m})$ on the competitive ratio of any on-line algorithm (preemptive or non-preemptive) for jobs with precedence constraints even when the running times are known. This matches the upper bound of Jaffe [11] who gave an approximation algorithm which can be implemented in an on-line environment. In fact, Davis and Jaffe [4] already gave a lower bound of $\Omega(\sqrt{m})$ on the competitive ratio for the case with no precedence constraints (i.e. batch-style scheduling) which obviously holds for the case of precedence constraints. However, our lower bound does not follow from their lower bound since their lower bound is valid only for unknown running times and no preemption.

We would like to emphasize that the precedence constraints are crucial for proving the lower bounds with known running times, since, otherwise, it becomes an off-line problem. Specifically, if the running times are known for the model without precedence constraints (i.e. batch-style scheduling) then all jobs are known in advance. Hence, one can get 1 competitive algorithm for the preemptive case ([10, 6]) and $1+\epsilon$ competitive algorithm for the non-preemptive case ([9]) (it becomes 1 if we allow exponential time algorithms). The precedence constraints are also crucial for proving the lower bounds with preemption even for unknown running times. Specifically, if we allow preemption then Shmoys, Wein and Williamson [13] showed an upper bound of $O(\log m)$ for the batch-style scheduling (i.e no precedence constraints). The upper bounds above for the model without precedence constraints should be contrasted with our result that implies that with precedence constraints one cannot get a better bound than $\Theta(\sqrt{m})$ even if the running times are known. Moreover, our lower bound holds for randomized preemptive online algorithm versus non-preemptive optimal off-line. It is worthwhile to mention that for the off-line version of our problem (i.e with precedence constraints) an $O(\log m)$ approximation algorithm is given in [3].

For the restricted assignment model we consider the algorithm Greedy which is described later. Azar et al [1] showed that for the case of no precedence constraints the Greedy algorithm for scheduling jobs one by one in the restricted assignment model achieves a competitive ratio of $O(\log m)$. In fact, the result in [1] is proved for scheduling over lists (i.e., scheduling jobs one by one). Nevertheless, their result immediately implies that Greedy is $O(\log m)$ competitive for scheduling jobs in the batch-style model with unknown running times. We show that if we allow consistent precedence constraints then the competitive ratio of the algorithm is still $O(\log m)$. We show that the algorithm is optimal up to a constant factor in this case by providing a lower bound of $\Omega(\log m)$ on the competitive ratio of any deterministic or randomized algorithm for scheduling jobs with restricted assignment and consistent precedence constraints. Our lower bound holds even for the known running times case and the upper bound does not use the running times. Moreover, the lower bound holds even for randomized preemptive algorithm versus non-preemptive optimal off-line while the upper bound holds for non-preemptive algorithm versus preemptive optimal off-line. We note that the precedence constraints are crucial for proving the lower bounds with known running times, since, otherwise, it becomes an off-line problem. In [1] there is a lower bound of $\Omega(\log m)$ for a model without precedence constraints but a job must be assigned immediately upon its arrival.

For general precedence constraints for the restricted assignment model we show a lower bound of m for the competitive ratio of any online algorithm ($\Omega(m)$ for randomized algorithms). This bound is easily matched by an algorithm which is m competitive. It is also easy to design an m competitive algorithm for the unrelated machines case. Recall that the unrelated machines case is a generalization of the restricted assignment model. Hence, the unrelated machines case is not of an interest since the best competitive ratio is m.

The Greedy algorithm. We adapt the Greedy algorithm "List", given by Graham [7] for identical machines, to the case of restricted assignment as follows.

Each time that a machine i becomes idle, assign to it a job j (if exists) such that $i \in M(j)$ and j has not been scheduled yet. Each time that a new job j arrives, assign it to an idle machine $i \in M(j)$ if exists. Note that Greedy is deterministic and does not use preemptions.

Randomized algorithms. To prove 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 $E(C_{on}/C_{opt}) \geq c$ for all deterministic on-line algorithms, then c is a lower bound on the competitive ratio of all randomized algorithms for the problem (see [2]). We will use only sequences for which C_{opt} is constant and thus in our case $E(C_{on}/C_{opt}) = E(C_{on})/C_{opt}$.

2 Scheduling on related machines

Theorem 2.1 *Any on-line algorithm for scheduling jobs with precedence constraints on m related machines has a competitive ratio of at least $\Omega(\sqrt{m})$. This is true even for randomized preemptive algorithms versus non-preemptive optimal off-line.*

Proof: We start by considering deterministic algorithms. We assume without loss of generality that $m-1$ is a square, $\sqrt{m-1} = r$. (Otherwise, we can restrict ourselves to the largest number which is at most m and of type r^2+1 for integer r and assume that all other machines have very small speed.) The set of machines consists of one fast machine of speed $r = \sqrt{m-1}$ (machine 1) and $m-1$ slow machines of speed 1 (machines $2, \ldots, m$). There are r phases of $r+1$ unit jobs each in the sequence. The sequence begins with $r+1$ independent unit jobs (phase 1). Next we define phase i, $2 \leq i \leq r$, the phase contains $r+1$ units jobs. Let b_{i-1} be a job in phase $i-1$ that finishes last by the on-line algorithm, then all jobs of phase i depend on b_{i-1}.

The on-line algorithm, by the definition of b_i, can start scheduling phase $i+1$ only after all jobs of phase i are completed. Since each phase consists of $r+1$ jobs, it is possible to use at most $r+1$ machines at any time. The $r+1$ fastest machines can process at most $2r$ unit jobs in one unit of time, and since the total running time of all jobs in one phase is $r+1$, each phase takes at least $(r+1)/(2r) \geq 1/2$ time units. Thus the total time to process all the sequence is at least $r((r+1)/(2r)) = (r+1)/2 = \Omega(\sqrt{m})$ (see Figure 1).

The optimal off-line algorithm assigns each b_i to the fast machine at time $(i-1)/r$, and thus the jobs of phase $i+1$ may be assigned at time i/r to machines $ir+2, \ldots, (i+1)r+1$ (for $0 \leq i \leq r-1$). The jobs of phase r would finish at time $(r-1)/r+1 \leq 2$ on the slow machines. The fast machine would finish at time 1 and thus $C_{opt} \leq 2$ (see Figure 2). The competitive ratio is $\Omega(\sqrt{m})$.

To extend the proof for randomized algorithms, b_i is chosen uniformly at random among all jobs of phase i. Clearly the optimal schedule remains the same. Next we evaluate the expected on-line schedule. The probability that the period of time starting from the arrival of phase i, till b_i is completed is at least T would be $(r+1-k)/(r+1)$ where k is the maximum number of jobs that it is possible to complete in a period of T units of time. For $T=(r+1)/(4r)$, it is possible to complete at most $\lfloor (r+1)/2 \rfloor$ jobs and thus the expectation of the period of time that passes from the arrival of b_i and till it is completed is at least $(r+1)/(8r) \geq 1/8$, and thus $E(C_{on}) = \Omega(\sqrt{m})$ and again the competitive ratio is $\Omega(\sqrt{m})$ as well.

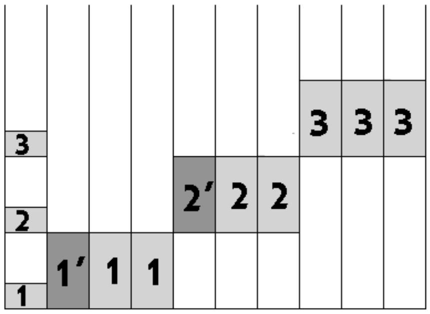

Figure 1: A possible on-line assignment in the proof of Theorem 2.1 for $m = 10$ and $r = 3$, where $C_{on} = 3$. In all figures the horizontal axis represents the machines, and the vertical axis represents time. Jobs marked by i belong to phase i, and the job marked i' is job b_i. In this figure the leftmost machine (machine 1) has speed 3 and all other machines have speed 1.

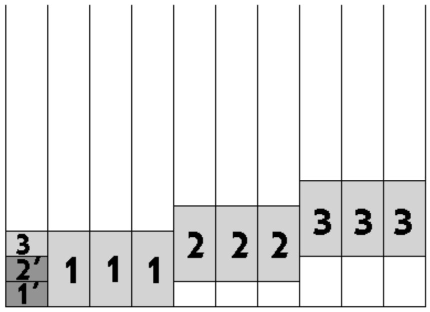

Figure 2: An optimal off-line assignment in the proof of Theorem 2.1 for $m = 10$ and $r = 3$. Machines have the same speeds as in Figure 1 (respectively) and $C_{opt} = 5/3$.

■

3 Restricted assignment with consistent precedence constraints

In this section we consider consistent precedence constraints for the restricted assignment model. Recall that precedence constraints are called consistent if for every j_1 which is a predecessor of j_2 we have $M(j_2) \subseteq M(j_1)$.

Theorem 3.1 *Any on-line scheduling algorithm for the restricted assignment model with consistent precedence constraints has a competitive ratio of at least* $\Omega(\log m)$*. This is true even for randomized preemptive algorithms versus non-preemptive optimal off-line.*

Proof: We assume without loss of generality that m is a power of 2, $m = 2^k$. (Otherwise we restrict ourselves to the largest number which is at most m and of type 2^k for an integer k and assume that no job can be assigned to the remaining machines.) The sequence consists of mN jobs where $N \geq 2\log_2 m = 2k$, the jobs belong to $k+1$ phases, where for $1 \leq i \leq k$ phase i contains $m(N+2-i)/2^i$ unit jobs, and phase $k+1$ contains $N - \log_2 m$ unit jobs. The jobs of phase i are restricted to machines $\{1, \ldots, 2^{k-i+1}\}$. We define the dependencies according to the behavior of the on-line algorithm. For $i = 1, \ldots, k$, let b_i be a job that finishes last of phase i, then all jobs of phase $i+1$ depend on b_i.

Since b_i is a job that finishes last at phase i, and all jobs of phase $i+1$ depend on it, then no jobs of phase $i+1$ are scheduled until all jobs of phase i are done. For $1 \leq i \leq k$, the jobs of phase i are restricted to $2^{k-i+1} = m/2^{i-1}$ machines, thus the period of time to finish all jobs of phase i is at least $(N+2-i)/2 = \Omega(N)$ (even with preemptions). Since there are $\Omega(\log m)$ phases, $C_{on} = \Omega(N \log m)$ (see Figure 3).

The optimal off-line algorithm schedules all b_i on the first machine, each b_i is scheduled at time $i-1$. The jobs of phase i are scheduled as follows: $m/2^{i-1}$ jobs are scheduled on machines $1, \ldots, m/2^{i-1}$ at time $i-1$, all the other jobs are scheduled from time i till time N on machines $m/2^i + 1, \ldots, m/2^{i-1}$. The jobs of phase $k+1$ are scheduled on machine 1 starting at time $\log_2 m$ (see Figure 4). We conclude that since $C_{opt} = N$, the competitive ratio is $\Omega(\log m)$.

To extend the proof for randomized algorithms we use the same sequence, but b_i is chosen uniformly at random among all jobs of phase i. Let P_i be the number of jobs that finish before b_i in phase i (jobs that finish at the same time are ordered arbitrarily). The period of time after the jobs of phase i become available and before the next phase can start is at least $(P_i+1)/2^{k-i+1}$. Since P_i gets the values $0, \ldots, (N+2-i)2^{k-i} - 1$ with equal probability,

$$E(P_i) = ((N+2-i)2^{k-i} - 1)/2 .$$

Hence,

$$\begin{aligned} C_{on} &\geq \sum_{i=1}^{k} E(P_i + 1)2^{-k+i-1} + N - \log_2 m \\ &\geq \sum_{i=1}^{k} (N + 2 - i)/4 = \Omega(N \log m) \ . \end{aligned}$$

Since $C_{opt} = N$ we conclude that the competitive ratio is $\Omega(\log m)$.

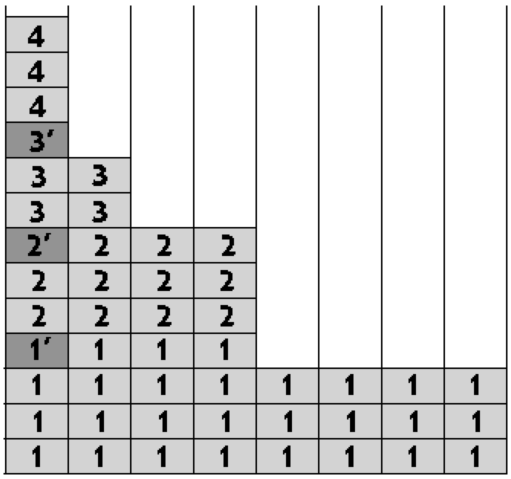

Figure 3: A possible on-line assignment in the proof of Theorem 3.1 for $m = 8$, $k = 3$, $N = 6$, where $C_{on} = 13$. All machines are identical.

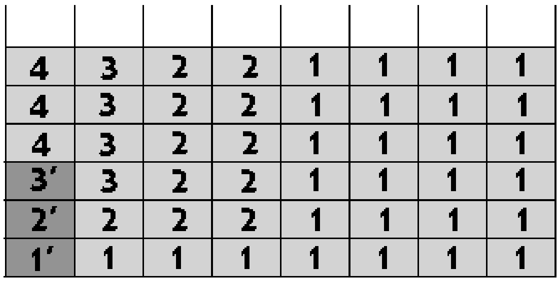

Figure 4: An optimal off-line assignment in the proof of Theorem 3.1 for $m = 8$, $k = 3$, $N = 6$ and $C_{opt} = 6$.

■

Theorem 3.2 *The competitive ratio of Greedy is $O(\log m)$ for the restricted assignment model with consistent precedence constraints.*

Proof: For machine i, let $A(i)$ be the set of jobs j that $i \in M(j)$. Denote the optimal off-line value by λ. We first prove the following Lemma:

Lemma 3.1 *The total idle time of Greedy on a machine i, from the beginning till the last job in $A(i)$ finishes its process (on any machine) is bounded by λ.*

Proof: For each machine i, we build a chain of jobs in which each job is dependent on the previous job, and each time i is idle, one of the jobs in the chain is running. Since the total running time of jobs in the chain is at most λ (the optimal off-line algorithm can not run more than one job of the chain simultaneously), the total idle time of machine i would be also bounded by λ. We build the chain from the top, starting from the last job in the chain. If there is no idle time on machine i, the chain is empty and the lemma follows. Otherwise, we start the chain with the job in $A(i)$ that finishes last, denote it by J_1. Assume that $J_1, \ldots, J_{q-1}$ are defined. If J_{q-1} has no predecessors, we finish the chain. Otherwise, let J_q be the predecessor of J_{q-1} that finishes last. Note that since all the chain consists of predecessors of J_1 and the precedence constraints are consistent, all the jobs in the chain are also in $A(i)$. Assume that i is idle at time t, and no job in the chain is running at time t. There is at least one job that finishes after time t (J_1 for example). Since there is no job of the chain running at time t, all these jobs start running after time t. Let J_r be the first job of the chain that starts running after time t. All the predecessors of J_r finish before time t thus since i is idle at t, J_r could be scheduled at time t or before. This is a contradiction to the definition of Greedy. ∎

Note that the idle time on each machine in Lemma 3.1 can be partitioned into two parts. The first is the idle time on a machine up to the completion of the last job that runs on this machine. The second is from that time on.

Lemma 3.2 *Let $l \geq 3\lambda$ be some time during the process of the algorithm. If the total running time of jobs (or parts of jobs) that run after time l is T_l then the total running time of jobs that run after time $l - 3\lambda$ is at least $2T_l$.*

Proof: Let $k_1 = \lceil \frac{T_l}{\lambda} \rceil$. The optimal off-line uses at least k_1 machines to run the jobs that the on-line runs after time l. Since the maximum running time is bounded by λ, these jobs start after time $l - \lambda$. For each machine i among the k_1 machines, there is a job that is allowed to be scheduled on it and is scheduled after time $l - \lambda$, thus machine i has at most λ idle time from time $l - 3\lambda$ till time $l - \lambda$. The total running time on i in this time period is at least λ. Summing for all machines the total running time is at least $k_1\lambda$, and adding the running times after time l we get a total of $k_1\lambda + T_l \geq T_l + T_l = 2T_l$ ∎

Now, we can complete the proof of the theorem. Let T be the total running time of all jobs, note that $T \leq m\lambda$. Let $k = \lfloor C_{on}/(3\lambda) \rfloor$. We can assume without loss of generality that $C_{on} \geq 3\lambda$. Hence $k \geq 1$. Note that the competitive ratio r satisfies $r = O(k)$. Let T_j be the total running time of jobs after time $C_{on} - 3j\lambda$. According to Lemma 3.2, T_k satisfies $T_k \geq 2^{k-1}T_1$ and according to Lemma 3.1, T_1 satisfies $T_1 \geq 2\lambda$, this is correct since there is at least one machine that finishes at time C_{on}, and since the idle time on this machine is bounded by λ, this machine worked at least for a period of time 2λ after time $C_{on} - 3\lambda$. Combining all observations together we get $m\lambda \geq T \geq T_k \geq 2^{k-1}T_1 \geq 2 \cdot 2^{k-1}\lambda$. Thus $k = O(\log m)$, and also $r = O(\log m)$.

■

4 Restricted assignment with general precedence constraints

In this section we consider the restricted assignment model with general precedence constraints between jobs.

Theorem 4.1 *Any on-line scheduling algorithm for restricted assignment model with general precedence constraints has the competitive ratio of at least m. This is true even for preemptive algorithms versus non-preemptive optimal off-line. Any randomized algorithm for the same problem has a competitive ratio of $\Omega(m)$. This is true even for randomized preemptive algorithms versus non-preemptive optimal off-line.*

Proof: We first prove a lower bound for the competitive ratio of deterministic algorithms, and later extend it to randomized ones. We build the sequence according to the behavior of the on-line algorithm. Let N be an integer $N \geq m$, The optimal cost for the sequence would be N. The sequence contains m phases, in each phase, all jobs are restricted to a single machine. Phase 1 contains N unit jobs which are restricted to machine 1. Let b_1 be the job from phase 1 that finishes last. We define the other phases recursively: In phase i ($i \geq 2$), there are $N - i + 1$ unit jobs which depend on the job b_{i-1}, and are restricted to machine i. We denote the job from phase i that finishes last by b_i.

The on-line does not schedule any job from phase $i + 1$ until all jobs of phase i are completed, because all jobs of phase $i + 1$ depend on b_i, thus the on-line has at most one working machine at a time (each job is restricted to a single machine) and the minimum possible on-line makespan is simply the sum of all running times: $C_{on} \geq \sum_{i=1}^{m}(N - i + 1) = m(N - m/2 + 1/2)$ (see Figure 5).

The optimal off-line algorithm assigns each b_i at time $i - 1$, and all other jobs of phase i are scheduled starting from time i, hence $C_{opt} = N$ (see Figure 6). The competitive ratio is at least $m - m^2/(2N) + m/(2N) = m - m(m-1)/(2N)$, for large values of N, this number approaches m.

To extend the proof to randomized algorithms we use a similar sequence, which also has m phases, where phase i contains $N - i + 1$ jobs that are restricted to machine i, but here the job b_i for $i = 1, \ldots, m - 1$ is chosen uniformly at random among all

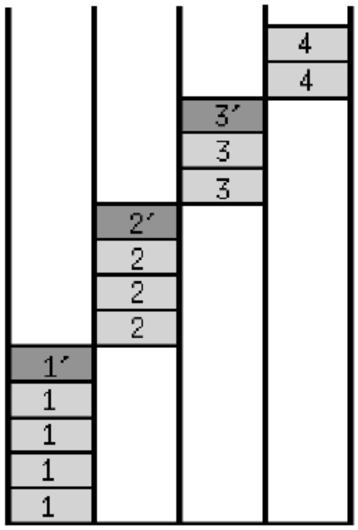

Figure 5: A possible on-line assignment in the proof of Theorem 4.1 for $m = 4$, $N = 5$, where $C_{on} = 14$.

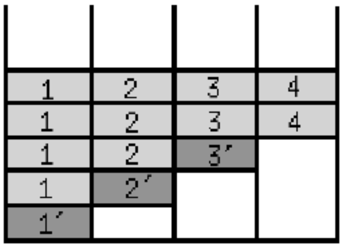

Figure 6: An optimal off-line assignment in the proof of Theorem 4.1 for $m = 4$, $N = 5$ and $C_{opt} = 5$.

jobs of phase i. Let P_i be the position of b_i, which is the number of jobs from phase i that were completed before b_i was completed. P_i can get the values $0, \ldots, N-i$, all with equal probabilities. For $i \geq 2$, the jobs of phase i are scheduled after at least $P_{i-1}+1$ jobs were completed at phase $i-1$ and thus $C_{on} = \sum_{i=1}^{m-1}(P_i+1)+N-m+1$. Thus

$$E(C_{on}) \geq \sum_{i=1}^{m-1}(E(P_i)+1) + N - m + 1 \ ,$$

since $E(P_i) = (N-i)/2$ we get

$$\begin{aligned} E(C_{on}) &\geq (m-1)(N/2+1) - \sum_{i=1}^{m-1} i/2 + N - m + 1 \\ &= mN/2 - N/2 + m - 1 - m(m-1)/4 + N - m + 1 \\ &= mN/2 + N/2 - O(m^2) \ . \end{aligned}$$

Since $C_{opt} = N$, the competitive ratio is at least $(m+1)/2 - O(m^2/N)$, for large values of N, the lower bound approaches $(m+1)/2 = \Theta(m)$.

Both lower bounds are valid even with preemptions since we only consider finishing times of jobs, and not starting times.

■

Theorem 4.2 *Greedy is m competitive for the restricted assignment model with precedence constraints.*

Proof: If all machines become idle, then there are no new jobs and the sequence is completed. Thus if $C_{on} = T$, then at any time between 0 and T there is at least one working machine. Hence, the sum of all processing times is at least T, and $C_{opt} \geq T/m$. Hence Greedy is m competitive. ■

We can easily provide m competitive algorithm also for unrelated machines. The algorithm Min assigns a job j to a machine i such that the running time of j on i is minimum over all i.

Theorem 4.3 *Min is m competitive for the unrelated machines model with precedence constraints.*

Proof: Since the running time that the optimal off-line uses to run each job is at least that of Min, we can imitate the proof of Theorem 4.2 ■

References

[1] Y. Azar, J. Naor, and R. Rom. The competitiveness of on-line assignments. *Journal of Algorithms*, 18(2):221–237, 1995. Also in *Proc. 3rd ACM-SIAM SODA*, 1992, pp. 203-210.

[2] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis.* Cambridge University Press, 1998.

[3] F.A. Chudak and D.B. Shmoys. Approximation algorithms for precedence constrained scheduling problems on parallel machines that run at different speeds. *Journal of Algorithms*, 30:323–343, 1999.

[4] E. Davis and J.M. Jaffe. Algorithms for scheduling tasks on unrelated processors. *J. Assoc. Comput. Mach.*, 28:712–736, 1981.

[5] L. Epstein. Lower bounds for on-line scheduling with precedence constraints on identical machines. In *1st Workshop on Approximation Algorithms for Combina-*

torial Optimization Problems (APPROX98), volume 1444 of *LNCS*, pages 89–98, 1998.

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

[7] R.L. Graham. Bounds for certain multiprocessor anomalies. *Bell System Technical Journal*, 45:1563–1581, 1966.

[8] R.L. Graham. Bounds on multiprocessing timing anomalies. *SIAM J. Appl. Math*, 17:416–429, 1969.

[9] D. Hochbaum and D. Shmoys. A polynomial approximation scheme for scheduling on uniform processors: Using the dual approximation approach. *SIAM Journal on Computing*, 17(3):539–551, 1988.

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

[11] J.M. Jaffe. Efficient scheduling of tasks without full use of processor resources. *Theoretical Computer Science*, 12:1–17, 1980.

[12] J. Sgall. On-line scheduling. In *A. Fiat and G. J. Woeginger, editors,* On-line Algorithms: The State of the Art, volume 1442 of *LNCS*, pages 196–231. Springer-Verlag, 1998.

[13] D. B. Shmoys, J. Wein, and D. P. Williamson. Scheduling parallel machines on line. *SIAM J. on Computing*, 24:1313–1331, 1995.