Optimal preemptive on-line scheduling on uniform processors with non-decreasing speed ratios *

Leah Epstein†

keywords: algorithms, scheduling.

Abstract

We study preemptive scheduling on uniformly related processors, where jobs are arriving one by one in an on-line fashion.

We consider the class of machine sets where the speed ratios are non-decreasing as speed increases. For each set of machines in this class, we design an algorithm of optimal competitive ratio. This generalizes the known result for identical machines, and solves other interesting cases.

1 Introduction

We consider on-line scheduling on m uniformly related machines. Jobs arrive on-line, and each job has to be assigned before the next job arrives. This scheduling model is called "scheduling jobs one by one" (see [10]). Preemption is allowed, hence each job may be cut into a few pieces. These pieces are to be assigned to possibly different machines, in non-overlapping time slots. (Non-preemptive algorithms are not allowed to cut the job and have to assign it continuously to one machine.)

*A preliminary version of this work appeared in the Proceedings of the 18th Annual Symposium on Theoretical Aspects of Computer Science (STACS'2001), pages 230-237.

†School of Computer and Media Sciences, The Interdisciplinary Center, P.O.B. 167, 46150 Herzliya, Israel. E-Mail: epstein.leah@idc.ac.il

Each job j is associated with a weight $w(j)$ and each machine i has a speed s_i. The processing time of a job (or a part of a job) of weight w, on machine i is w/s_i. The machines are sorted so that $s_i \leq s_{i+1}$ for $1 \leq i \leq m-1$ and $s_m = 1$. The last condition is general since it is possible to scale any set of speeds and job weights into this form. The load of machine i, L_i, is equal to the processing times of all parts of jobs assigned to machine i, on this machine, divided by its speed s_i. The goal of an algorithm is to minimize the makespan, which is the maximum load on any machine.

The quality of an on-line algorithm is measured by the competitive ratio that is the worst case ratio between C_{on} which is the cost (the makespan, in our case) of the on-line algorithm and C_{opt}, which is the cost of an optimal off-line algorithm which knows all the sequence in advance.

Previous work: Chen, Van Vliet and Woeginger gave a preemptive optimal algorithm for identical machines [4]. They show that the best competitive ratio for identical machines is $m^m/(m^m-(m-1)^m)$. A lower bound of the same value on the competitive ratio of non-preemptive randomized algorithms is also known. The proofs use similar sequences as the ones in [4] and were given independently by [3] and [9]. However, no optimal non-preemptive randomized algorithm is know for $m > 3$. (For m=2, such an optimal algorithm is given in [1].)

Preemptive scheduling on two related machines was studied independently by [5] and by [12]. Both papers show that the optimal competitive ratio is $1+s_1/(s_1^2+s_1+1)$.

Preemptive scheduling on related machines was also considered by Epstein and Sgall [6]. That paper gives a constant competitive algorithm for any m and set of speeds. The paper also gives a lower bound of 2 on the competitive ratio of any algorithm with an unbounded number of machines and specific lower bounds for constant values of m. Those lower bounds are valid for randomized preemptive or non-preemptive algorithms.

Our results: In this paper we solve the case of non-decreasing speed ratios, i.e. $\frac{s_{i-1}}{s_i} \leq \frac{s_i}{s_{i+1}}$ for $2 \leq i \leq m-1$. Note that both the case of identical machines, and the case of two related machines fit into this special case. For a given set of speeds, let $x = \sum_{i=1}^{m} s_i / (\sum_{i=1}^{m} s_i - 1)$. Note that $\sum_{i=1}^{m} s_i = \frac{x}{x-1}$ and $\sum_{i=1}^{m} s_i - 1 = \frac{1}{x-1}$. We prove the following theorem:

Theorem 1 *Given a set of related machines with non-decreasing speed ratios. The best competitive ratio for preemptive on-line scheduling, is* $\frac{x^m}{(x-1)\sum_{i=1}^{m} s_i x^{i-1}}$. *This holds for deterministic and randomized algorithms.*

To prove the theorem, we give an algorithm of optimal competitive ratio for every set of speeds. Specifically, we design a deterministic preemptive algorithm of competitive ratio $\frac{x^m}{(x-1)\sum_{i=1}^{m} s_i x^{i-1}}$ and a matching lower bound. The lower bounds are valid for deterministic and randomized preemptive algorithms. This ratio is at most 2 for all sets of speeds with non-decreasing speed ratios.

As mentioned above, all previous results are a special case of this pattern. Hence, we generalize all known results where matching upper and lower bounds are known. Many ideas of our results are based on [4]. The definition and the proof of the algorithm and also the structure of the lower bound are similar. However, it was not clear which special cases of related machines can be solved by similar methods, and in what way, the algorithm could be appropriately generalized. Our lower bounds are the general case of the lower bound in [6] for unbounded m. Even though our result does not hold for non-preemptive algorithms, in Section 3 we mention some cases where it holds, and one of these cases is an exponential set of speeds ($s_i = y^{m-i}$ for some $0 < y < 1$) which is also used in [6]. The tight competitive ratio in this case is $\frac{x^m(x-y)}{(x-1)(x^m-y^m)}$.

Note that Vestjens [11] was the first to identify the importance of non-decreasing speed ratios for related machines. He studied a preemptive on-line scheduling model where jobs arrive over time, instead of one by one. It is surprising that the same property arises again in a different on-line scheduling model. Vestjens showed that for his model, an algorithm with competitive ratio 1, which uses a finite number of preemptions can be given if and only if speed ratios are non-decreasing.

We start the paper with definitions and proofs of the optimal algorithms and prove the lower bounds in Section 3.

2 Algorithms

We describe the preemptive algorithm. Note that it is easy to compute the optimal preemptive off-line makespan at every step. The formula was given by [8] and by [7]. The optimal load is the maximum of the following m values: the total weight of all jobs divided by the sum of speeds, and for $1 \le j \le m-1$, the sum of weights of the largest j jobs divided by the sum of largest j speeds of machines.

Our algorithm, similarly to [4] tries to maintain a ratio of x between loads of subsequent machines.

We use the following notations; the load of machine l after the arrival of t jobs is denoted by L_l^t. The optimal off-line makespan at that time is denoted by C_{opt}^t and the sum of weights of the first t jobs is denoted by W^t.

Let

$$r = \frac{x^m}{(x-1)\sum_{i=1}^{m} s_i x^{i-1}} \quad .$$

The algorithm maintains the following three invariants. These invariants are a generalization of the invariants defined for identical machines in [4].

- At any time t,
  $L_1^t \le L_2^t \le \cdots \le L_m^t$.
- At any time t,
  $L_m^t \le r \cdot C_{opt}^t$.
- At any time t, for every $1 \le k \le m$
  $\sum_{i=1}^{k} s_i L_i^t \le \frac{\sum_{i=1}^{k} s_i x^{i-1}}{\sum_{i=1}^{m} s_i x^{i-1}} \cdot W^t$

A new job J_{t+1} (which arrives at time $t+1$) is assigned as follows. The new optimal off-line makespan is computed using its weight $w(J_{t+1})$. Then the following intervals are reserved. On machine m, the interval:

$$I_m = [L_m^t \ , \ r \ C_{opt}^{t+1}] \quad ;$$

and on machine j $(1 \le j \le m-1)$, the interval $I_j = [L_j^t \ , L_{j+1}^t]$. Those intervals are disjoint. The intervals relate to load and not to weight, the weight that can be assigned on I_j $(1 \le j \le m-1)$ is $s_j(L_{j+1}^t - L_j^t)$.

To assign J_{t+1}, go from I_m to I_1, putting a part of the job, as large as possible in each interval, until all the job is assigned. After the assignment there will be some fully occupied intervals $I_{z+1}, \cdots, I_m$, some empty intervals $I_1, \cdots, I_{z-1}$ and a partially or fully occupied interval I_z.

Next, we show that it is always possible to partition a job among those intervals. For convenience define $s_0 = 0$ and $s_{m+1} = 1$. Then $s_{i-1}/s_i \le s_i/s_{i+1}$ holds for all $1 \le i \le m$.

Lemma 1 *If the invariants are fulfilled at step t, then the reserved intervals are sufficient to assign J_{t+1}.*

Proof: The total weight that can be assigned to all intervals is

$$\begin{aligned} A &= (r\ C_{opt}^{t+1} - L_m^t)s_m + \sum_{j=1}^{m-1}(L_{j+1}^t - L_j^t)s_j \\ &= r\ C_{opt}^{t+1} + \sum_{j=1}^{m}(s_{j-1} - s_j)L_j^t \\ &= r\ C_{opt}^{t+1} + \sum_{j=1}^{m}\left(\frac{s_{j-1}}{s_j} - \frac{s_j}{s_{j+1}}\right)\sum_{i=1}^{j} s_i L_i^t \end{aligned}$$

Since $s_{j-1}/s_j \le s_j/s_{j+1}$ we can use the third invariant for each value of j and get that the above is at least

$$\begin{aligned} A &\ge \frac{1}{\sum\limits_{i=1}^{m} s_i x^{i-1}}\left[\frac{x^m}{x-1}C_{opt}^{t+1} + W^t \sum_{j=1}^{m}\left(\frac{s_{j-1}}{s_j} - \frac{s_j}{s_{j+1}}\right)\sum_{i=1}^{j} s_i x^{i-1}\right] \\ &= \frac{1}{\sum\limits_{i=1}^{m} s_i x^{i-1}}\left[\frac{x^m}{x-1}C_{opt}^{t+1} + \left((x-1)\sum_{j=1}^{m} s_j x^{j-1} - x^m\right)W^t\right] \end{aligned}$$

We consider two cases:

1. $w(J_{t+1}) \ge \frac{W^{t+1}}{\sum_{i=1}^{m} s_i}$ and then $C_{opt}^{t+1} \ge w(J_{t+1})$.

2. $w(J_{t+1}) \le \frac{W^{t+1}}{\sum_{i=1}^{m} s_i}$ and then $C_{opt}^{t+1} \ge W^{t+1}/\sum_{i=1}^{m} s_i$.

We show that the assignment is successful in both cases.

Case 1:

Since the term multiplied by W^t is non-positive we can substitute $W^t \le w(J_{t+1})(\sum_{i=1}^{m} s_i - 1)$ and get that

$$\begin{aligned} A \ &\ge \ \frac{1}{\sum_{i=1}^{m} s_i x^{i-1}} \Bigg[\frac{x^m}{x-1} w(J_{t+1}) \\ &+ \ w(J_{t+1})(\sum_{i=1}^{m} s_i - 1)((x-1)\sum_{i=1}^{m} s_j x^{j-1} - x^m) \Bigg] \end{aligned}$$

simplifying this gives

$$A \ge w(J_{t+1}).$$

Case 2:

In this case, $C_{opt}^{t+1} \ge \frac{w(J_{t+1}) + W^t}{\sum_{i=1}^{m} s_i}$. Substituting this we get that the term multiplied by W^t is $\frac{x^m}{(x-1)\sum_{i=1}^{m} s_i} - x^m + (x-1)\sum_{i=1}^{m} s_i x^{i-1}$, which is non-negative. By using $W^t \ge w(J_{t+1})(\sum_{i=1}^{m} s_i - 1)$ we get

$$\begin{aligned} A \ &\ge \ \frac{w(J_{t+1})}{\sum_{i=1}^{m} s_i x^{i-1}} \cdot \Bigg[\frac{1}{\sum_{i=1}^{m} s_i} \left(\frac{x^m}{x-1} + \frac{x^m}{x-1} \cdot (\sum_{i=1}^{m} s_i - 1) \right) + \\ &(\sum_{i=1}^{m} s_i - 1)\left((x-1)\sum_{i=1}^{m} s_j x^{j-1} - x^m \right) \Bigg] \end{aligned}$$

simplifying this also gives $A \ge w(J_{t+1})$. ■

To complete the proof of the algorithm, we need to show that all invariants are kept after an assignment of a job. This is clear for the first two invariants, from the definition of the algorithm.

Lemma 2 *If the invariants are fulfilled after step t, then they are also satisfied after step $t+1$*

This would be sufficient since at the start, all loads are zero.

Proof: We only need to show that the third invariant holds for every $1 \le k \le m$.

According to the definition of the algorithm, there exists a machine z such that for $i < z$, $L_i^{t+1} = L_i^t$, for $z < i \le m$, $L_i^{t+1} = L_{i+1}^t$, and $L_z^t < L_z^{t+1} \le L_{z+1}^t$ (for convenience let $L_{m+1}^t = rC_{opt}^{t+1}$).

If $k < z$, then

$$\sum_{i=1}^{k} s_i L_i^{t+1} = \sum_{i=1}^{k} s_i L_i^t \le \frac{\sum_{i=1}^{k} s_i x^{i-1}}{\sum_{i=1}^{m} s_i x^{i-1}} W^t \le \frac{\sum_{i=1}^{k} s_i x^{i-1}}{\sum_{i=1}^{m} s_i x^{i-1}} W^{t+1} \quad .$$

If $k \ge z$ then we need to show

$$\sum_{i=k+1}^{m} s_i L_i^{t+1} \ge \frac{\sum_{i=k+1}^{m} s_i x^{i-1}}{\sum_{i=1}^{m} s_i x^{i-1}} W^{t+1} \qquad (1)$$

Since $k \ge z$, $L_i^{t+1} = L_{i+1}^t$ and the left hand size is equal to

$$\sum_{i=k+2}^{m} s_{i-1} L_i^t + rC_{opt}^{t+1} = rC_{opt}^{t+1} + \sum_{i=k+2}^{m} s_i L_i^t \left(\frac{s_{k+1}}{s_{k+2}}\right) + \sum_{j=k+3}^{m} \left(\sum_{i=j}^{m} s_i L_i^t\right) \left(\frac{s_{j-1}}{s_j} - \frac{s_{j-2}}{s_{j-1}}\right)$$

since $\frac{s_{j-1}}{s_j} - \frac{s_{j-2}}{s_{j-1}} \ge 0$ for all j, we can use the invariants of step t, and get that this value is at least

$$\frac{1}{\sum_{i=1}^{m} s_i x^{i-1}} \left[\frac{x^m}{x-1} C_{opt}^{t+1} + \left(\sum_{i=k+1}^{m-1} s_i x^i\right) W^t \right]$$

Let $w(J_{t+1}) = \mu W^{t+1}$, then $C_{opt}^{t+1} = \max\left\{\mu, \frac{1}{\sum_{i=1}^{m} s_i}\right\} W^{t+1}$ and $W^t = (1-\mu) W^{t+1}$. Simple calculations show that inequality (1) holds. ∎

3 Matching lower bounds

To prove a matching lower bound, we use the following lemma, given in [6]. This lemma gives a general method of proving lower bounds for preemptive algorithms and for randomized algorithms.

Lemma 3 *Consider a sequence of at least m jobs, where $J_0, J_1, \cdots, J_{m-1}$ are the last m jobs, Let $C_{opt}(J_i)$ be the preemptive optimal off-line cost after the arrival of J_i.*

The competitive ratio of any preemptive randomized on-line algorithm is at least

$$W / \left(\sum_{i=1}^{m} s_i C_{opt}(J_{i-1}) \right)$$

where W is the total weight of all jobs in the sequence.

Note that if we consider non-preemptive optimal off-line costs, the same expression lower bounds the competitive ratio of non-preemptive randomized algorithms.

Before constructing the lower bound sequence, we introduce the following auxiliary claim.

Claim 1 *For each $1 \leq k \leq m-1$*

$$\sum_{i=1}^{k} x^{i-1} / \left(\sum_{i=1}^{k} s_{m-k+i} \right) \leq x^{k-1} \ .$$

Proof: Let j be a maximum index such that $s_j < \frac{1}{x^{m-j}}$. If no such index exists then $j = 0$. Note that $j < m$ since $s_m = 1$. Hence for $k > j$, $s_k \geq \frac{1}{x^{m-k}}$. We consider two cases.

Case 1: $j < m - k + 1$

Then $x^{k-1} \left(\sum_{i=1}^{k} s_{m-k+i} \right) \geq x^{k-1} \sum_{i=1}^{k} \frac{1}{x^{k-i}} = \sum_{i=1}^{k} x^{i-1}$.

Case 2: $j \geq m - k + 1$

Since $\frac{s_j}{s_{j+1}} < \frac{1}{x^{m-j}} \cdot \frac{x^{m-j-1}}{1} = \frac{1}{x}$ then by induction for all $p \leq j$, $s_p \leq \frac{1}{x^{m-p}}$. Hence

$$\begin{aligned}
\sum_{i=1}^{k} s_{m-k+i} &= \sum_{i=1}^{m} s_i - \sum_{i=1}^{m-k} s_i \\
&\geq \frac{x}{x-1} - \sum_{i=1}^{m-k} \frac{1}{x^{m-i}} \geq \frac{x}{x-1} - \frac{1}{x^{k-1}} \cdot \frac{1}{x-1} \\
&\geq \frac{1}{x^{k-1}} \sum_{i=1}^{k} x^{i-1} \quad .
\end{aligned}$$

■

Next, we construct the lower bound sequence. The construction is similar to the proofs in [3, 4, 9].

The sequence starts with an amount of $\sum_{i=1}^{m-1} s_i - 1$ of very small jobs (sand). These jobs are followed by $m-1$ jobs $J_1, \cdots, J_{m-1}$ where $W(J_i) = x^{i-1}$. The sequence is built so that the following lemma holds:

Lemma 4 *The preemptive optimal off-line cost after the arrival of i big jobs is x^{i-1}, this is true for $i = 0, \cdots, m-1$.*

Proof: The lemma is satisfied for $i = 0$, since splitting the sand evenly gives

$$C_{opt} = \frac{\sum_{i=1}^{m} s_i - 1}{\sum_{i=1}^{m} s_i} = \frac{1}{x} \quad .$$

For $i > 0$ let W^i be the total weight of jobs arriving no later than J_i.

C_{opt} after the arrival of J_i is the maximum between $W^i / \sum_{i=1}^{m} s_i$ and $\max_{1 \le k \le i} \left(\sum_{p=k}^{i} W(J_p) \right) / \left(\sum_{p=m+i-k}^{m} s_p \right)$. The first value is x^{i-1}.

According to the Claim 1, each term in the second value is at most x^{i-1}. This proves the lemma.

■

The proof of the following theorem follows from Lemma 3 and Lemma 4.

Theorem 2 *The competitive ratio of any preemptive randomized and any preemptive deterministic on-line algorithm is at least*

$$x^m / \left((x-1) \sum_{i=1}^{m} (s_i x^{i-1}) \right)$$

Note that if only case 1 of Claim 1 occurs, i.e. for all $i \ge 2$, $s_i \ge \frac{1}{x^{m-i}}$, then Lemma 4 holds also for the case where the optimal off-line algorithm does not use preemptions, and then the lower bound is valid for randomized non-preemptive on-line algorithms.

This is the case for two basic sets of machines.

1. All machines but the slowest have speed 1. The lower bound value in this case is $x^m / (x^m - 1 + (x-1)(s_1 - 1))$. This gives the result for m identical machines for

$s_1 = 1$ and for $m-1$ identical machines for $s_1 = 0$. (see [4]).

2. Machines speeds are powers of some number $0 < y < 1$, i.e. $s_i = y^{m-i}$. In this case $x = \frac{1-y^m}{y-y^m}$ hence $y \geq \frac{1}{x}$. The lower bound is $x^m(x-y)/((x-1)(x^m - y^m))$ which tends to $(y+1)$ as m goes to infinity. This lower bound is given in [6].

4 Conclusions and open problems

We have given optimal algorithms for a class of uniformly related machines. It would be interesting to give optimal algorithms for other classes. It an open problem whether there is a speed pattern where $\frac{x^m}{(x-1)\sum_{i=1}^{m} s_i x^{i-1}}$ is not the best possible competitive ratio.

The problem of computing the overall competitive ratio is also interesting. The best competitive ratio for related machines in general is unknown. There is a large gap between the lower bound of 2 given in [6], and the algorithm given there which has competitive ratio above 20. (The upper bound can be improved to 11.656 by using the algorithm of [2] and a claim from [13]).

References

[1] Y. Bartal, A. Fiat, H. Karloff, and R. Vohra. New algorithms for an ancient scheduling problem. *J. Comput. Syst. Sci.*, 51(3):359–366, 1995.

[2] P. Berman, M. Charikar, and M. Karpinski. A note on on-line load balancing for related machines. In *5th annual Workshop on Algorithms and Data Structures*, 1997.

[3] B. Chen, A. van Vliet, and G. J. Woeginger. Lower bounds for randomized online scheduling. *Information Processing Letters*, 51:219–222, 1994.

[4] B. Chen, A. van Vliet, and G. J. Woeginger. An optimal algorithm for preemptive on-line scheduling. *Operations Research Letters*, 18:127–131, 1995.

[5] L. Epstein, J. Noga, S.S. Seiden, J.Sgall, and G.J. Woeginger. Randomized online scheduling on two uniform machines. In *Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 317–326, 1999. To appear in Journal of Scheduling.

[6] L. Epstein and J. Sgall. A lower bound for on-line scheduling on uniformly related machines. *Oper. Res. Lett.*, 26(1):17–22, 2000.

[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,* On-line Algorithms: The State of the Art, volume 1442 of *LNCS*, pages 196–231. Springer-Verlag, 1998.

[11] A. P. A. Vestjens. Scheduling uniform machines on-line requires nondecreasing speed ratios. *Math. Programming*, 82(2):225–234, 1998.

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

[13] G. J. Woeginger. A comment on scheduling on uniform machines under chain-type precedence constraints. 26(3):107–109, 2000.