Accepted Manuscript

An improved algorithm for online machine minimization

Yossi Azar, Sarel Cohen

PII: S0167-6377(17)30392-9
DOI: https://doi.org/10.1016/j.orl.2017.11.013
Reference: OPERES 6309

To appear in: *Operations Research Letters*

Received date : 17 July 2017
Revised date : 20 November 2017
Accepted date : 20 November 2017

Please cite this article as: Y. Azar, S. Cohen, An improved algorithm for online machine minimization, *Operations Research Letters* (2017), https://doi.org/10.1016/j.orl.2017.11.013

An Improved Algorithm for Online Machine Minimization

Yossi Azar
Tel-Aviv University
Tel-Aviv, Israel
azar@tau.ac.il

Sarel Cohen
Tel-Aviv University
Tel-Aviv, Israel
sarelcoh@post.tau.ac.il

ABSTRACT

The online machine minimization problem seeks to design a preemptive scheduling algorithm on multiple machines — each job j arrives at its release time r_j, has to be processed for p_j time units, and must be completed by its deadline d_j. The goal is to minimize the number of machines the algorithm uses. We improve the $O(\log m)$-competitive algorithm by Chen, Megow and Schewior (SODA 2016) and provide an $O(\frac{\log m}{\log\log m})$-competitive algorithm.

KEYWORDS

Scheduling, Online Machine Minimization, Competitive Ratio, Analysis of Algorithms

1 INTRODUCTION

The machine minimization problem is to schedule a set of jobs with specified time intervals on the minimum number of machines required. Each job j arrives at its release time r_j, has to be processed for p_j time units, and must be completed by its deadline d_j. The goal is to minimize the number of machines the algorithm uses to ensure all jobs are processed by their deadline. In the preemptive model each processed job may be stopped and resumed later, possibly on a different machine. In the online setting, the jobs arrive online and the attributes (r_j, p_j, d_j) of the job are known to the algorithm at the release time r_j.

There are various well-known algorithms for scheduling jobs on machines. For example, EDF (earliest deadline first) schedules the jobs currently in the system which have the earliest deadlines. Another algorithm is LLF (least laxity first) schedules the jobs currently in the system according to their laxities (the jobs with the least laxities are scheduled first). At time t, a job j has laxity $\ell_j(t) = (d_j - t - p_j(t))$ where $p_j(t)$ is the remaining processing time of job j at time t.

Let m be the minimum number of machines required in order to ensure all jobs are processed by their deadline in the offline setting, when all the jobs and their attributes (r_j, p_j, d_j) are known in advance. In the uniprocessor case ($m = 1$), if there exists any feasible schedule, then both EDF ([9]) and LLF ([8]) find feasible solutions on a single machine. In the multiprocessor case ($m \geq 2$), any online algorithm must use more than m machines on some inputs ([8]).

Let $P_{\max}$ be the maximum processing time of a job and $P_{\min}$ be the minimum processing time of a job. Phillips, Stein, Torng, and Wein proved in [18] that LLF is $O(\log \frac{P_{\max}}{P_{\min}})$-competitive. They also showed a lower bound of 5/4 on the competitive ratio for any deterministic algorithm, leaving a huge gap of $O(\log \frac{P_{\max}}{P_{\min}})$ on the competitive ratio of LLF. They also show that EDF does not improve the competitive ratio by giving an $\Omega(\log \frac{P_{\max}}{P_{\min}})$ lower bound on EDF.

Nearly two decades later, Chen, Megow and Schewior (SODA 2016) [4] were the first to significantly improve the competitive ratio. They presented an algorithm which is $O(\log m)$-competitive. In particular, for fixed m this yields a constant competitive algorithm. They also showed that a variant of their algorithm is constant competitive when all jobs have processing time windows that are either laminar or agreeable. Let $I(j) = [r_j, d_j)$ be the time window of job j. In laminar instances, any two jobs j and j' with $I(j) \cap I(j') \neq \emptyset$ satisfy $I(j) \subseteq I(j')$ or $I(j') \subseteq I(j)$. In agreeable instances, $r_j < r_{j'}$ implies $d_j \leq d_{j'}$ for any two jobs j and j'. They claim that they don't know if the analysis of their algorithm is tight for the general case, and ask whether one can show a better analysis which improves their competitive ratio. Here we improve their analysis by a $\log\log m$ factor. Following our result, [14] recently improved the competitive ratio for online machine minimization to $O(\log\log m)$.

It is not known if the online machine minimization problem admits an $O(1)$-competitive algorithm or not, which is an important open problem in this field [4, 18, 19].

1.1 Our Contribution

As described above, it is a wide open question to narrow the gap between the $O(\log m)$ upper bound and the constant lower bound for the machine minimization problem. We go one step further in narrowing this gap. By giving stronger lower and upper bounds on the claims in [4], and by considering a better combination of EDF and their algorithm, we are able to improve the competitive ratio to $O(\frac{\log m}{\log\log m})$.

An extended abstract of this paper appeared in [1].

1.2 Related Work

Recall that Phillips *et al.* [18] show that LLF is $O(\log \frac{P_{\max}}{P_{\min}})$-competitive, and Chen *et al.* [4] present an $O(\log m)$-competitive algorithm.

If preemption is not allowed, then any online algorithm is $\Omega(n)$-competitive (See [21]). Note that for the special case of unit processing times, preemptions are not required and the optimal non-preemptive online algorithm has a competitive ratio of $e \approx 2.72$ (See [2, 10]). This model has been studied in various papers [10, 15, 16, 21, 22] and also in the context of energy minimization [2].

Job migration is the ability to start processing a job on a machine, and then preempt it and process it later, possibly on a different machine. Chen, Megow and Schewior prove in [5] that migration significantly affects machine minimization. Specifically, the number of machines required in a non-migrative solution is unbounded as a function of m (the number of machines required in a migrative solution).

Another variant, also studied by Phillips et al., considers the setting where no additional machines are used (only m machines are allowed), but instead each machine is faster by a speedup factor.