Journal of Scheduling
https://doi.org/10.1007/s10951-019-00629-3

An optimal online algorithm for scheduling with general machine cost functions

Islam Akaria[1] · **Leah Epstein**[1]

Abstract
We present two deterministic online algorithms for the problem of scheduling with a general machine cost function. In this problem, every machine has a cost that is given by a nonnegative cost function, and the objective function is the sum of makespan and the cost of the purchased machines. In previous work by Imreh, it was shown that no deterministic algorithm has competitive ratio below 2, and an algorithm whose competitive ratio does not exceed $(3+\sqrt{5})/2 \approx 2.618$ was presented. Our first algorithm imitates an optimal offline solution with respect to the number of machines used, and we show that its competitive ratio is exactly 2.5. Then, we modify our algorithm by imitating a preemptive optimal offline solution instead of a non-preemptive one. This leads to the design of a 2-competitive algorithm, which is the best possible.

Keywords Online scheduling · Competitive ratio · Machine cost · Preemptive scheduling

1 Introduction

In online machine scheduling problems, usually a fixed set of machines is given to the algorithm to be used for scheduling arriving jobs. It is often assumed that the provided machines do not incur any cost. Here, we study an online scheduling problem where it is the role of the algorithm to decide how many machines it will use, where the number of machines can be increased by the algorithm at any time. The goal of such studies is to design a more realistic model and to find the effect of this kind of modification of the model on basic scheduling problems. Online scheduling with machine cost was introduced by Imreh and Noga (1999).

Before considering scheduling models with machine cost, we define classic variants and in particular, those without machine cost.

Online scheduling on identical machines In this basic problem, a set of m machines is given to the algorithm. Jobs are presented one by one or over a list, and the algorithm assigns each job completely and irrevocably prior to the arrival of the next job. Every machine can run at most one job at each time, and every job can be run on at most one machine at each time. In the non-preemptive variant of our model, every job selects one machine where it should be executed continuously. In the preemptive variant, a job can be run on different machines at different times, and it does not necessarily have to be run continuously. Thus, in the non-preemptive case, it is sufficient to select a machine for every job. (And we assume that jobs are assigned to run there without idle time starting at time zero.) In the preemptive case, every job may be split into a finite number of parts, each to be assigned to a time slot on some machine, under the condition that the parts cannot run simultaneously. The objective is to minimize the makespan, which is the largest completion time (or load) of any machine or equivalently, the maximum completion time over the jobs. These problems (with fixed numbers of machines) were studied in the online environment and the offline environment (McNaughton 1959; Graham 1966; Hochbaum and Shmoys 1987; Chen et al. 1995; Fleischer and Wahl 2000). The offline non-preemptive variant is NP-hard, while the offline preemptive one has a polynomial time algorithm by McNaughton (1959) (which computes an optimal solution).

On scheduling with machine cost Unlike standard online scheduling problems, no machines are initially provided to

Partially supported by a grant from GIF—the German-Israeli Foundation for Scientific Research and Development (Grant Number I-1366-407.6/2016).

✉ Leah Epstein
lea@math.haifa.ac.il

Islam Akaria
islam.akaria@gmail.com

[1] Department of Mathematics, University of Haifa, Haifa, Israel

Published online: 09 December 2019

the algorithm. A nonnegative function is given in advance (by an oracle), representing the cost of buying each machine. When a job is presented, the algorithm may purchase any number of new machines. The objective is to minimize a function that is the sum of the makespan plus the cost of the machines ever purchased by the algorithm. Once the algorithm has a machine, it can schedule any new job on this machine (but previously assigned jobs cannot be reassigned). This problem is called *Scheduling with machine cost* (SMC). Here and in most previous work, SMC is investigated in the online list model where the jobs arrive one by one, and the decision maker has to schedule every job immediately in the sense that it is assigned irrevocably to the exiting machines without any information about further jobs, exactly as in the basic variant defined above. At each step, the algorithm may buy any number of machines, and then, it schedules the new job. We allow an algorithm to buy more than one machine and to schedule the new job on any machine it has ever bought and not only a new machine, even though in such cases (where a newly bought machine does not receive a job or a part of it just after it is bought), the purchase of new machines can be postponed.

As we study online algorithms for which it is generally not possible to obtain an optimal solution, we use competitive analysis for measuring quality of algorithms. An online scheduling algorithm is C-competitive if its cost never exceeds C times the optimal cost (for the same input), i.e., the cost of an optimal offline solution. The competitive ratio of an algorithm is the infimum C for which it is C-competitive. In other words, the competitive ratio of a given algorithm A for an input I is the ratio between the cost of A for I and the optimal cost for I, and the competitive ratio of A is the supremum C over the competitive ratios for all possible inputs.

In the study of Imreh and Noga (1999), the cost of every machine is defined to be 1, and therefore, the cost of k machines is simply k. In their work, they showed that the best possible competitive ratio for SMC is in the interval $[4/3, (1+\sqrt{5})/2 \approx 1.618]$. After this work, there was a sequence of improvements for the case of unit machine costs. An improved algorithm with competitive ratio $2(\sqrt{6}+3)/5 \approx 1.5798$ was presented by Dósa and He (2004). Dósa and Tan (2010) designed a new algorithm with competitive ratio $(2+\sqrt{7})/3 \approx 1.5486$ and showed a lower bound of $\sqrt{2}$ on the competitive ratio of the non-preemptive variant. For the preemptive variant, it was shown by Jiang and He (2005) that the lower bound of $\frac{4}{3}$ still holds, and an algorithm whose competitive ratio does not exceed 1.3798 was designed. In the algorithm of Imreh and Noga (1999), the main idea is to buy a new machine when the total size of jobs increases to some pre-defined value. Those values are based on powers of 2 of integers. Jobs were scheduled by the simple greedy approach of Graham (1966), under which every job is assigned to the least loaded (currently existing) machine. In the improved algorithms, not only the total size of jobs is used for finding the required number of machines, but also the maximum size of any job is taken into account. A variant with rejection (under penalties) was studied too (Dósa and He 2006; Nagy-György and Imreh 2007). Other variants such as semi-online algorithms, randomized online algorithms, and offline variants were studied (Seiden 2000; He and Cai 2002; Jiang and He 2006; Leung et al. 2012; Dósa and Imreh 2013).

Scheduling with general machine cost functions Imreh (2009) considered the problem with general cost function. In this variant, the cost of the machines can be arbitrary, and it is defined by a non-decreasing machine cost function denoted by $c(m)$ (where we assume that we can compute it exactly using an oracle). The value $c(m)$ is the cost of purchasing the first m machines, i.e., the cost of the mth machine ever purchased is $c(m)-c(m-1)$. This problem models a situation where new machines require space, and the cost of adding a new machine depends on how much empty space there is, and whether a new room is needed for the new machine. He analyzed an algorithm from a class of algorithms defined in Imreh and Noga (1999) and showed that the competitive ratio of that algorithm is $(3+\sqrt{5})/2 \approx 2.618$. He also considered a special case with small jobs where the processing time of a job cannot be larger than the minimal cost of any machine. For this case (only), he showed a 2-competitive algorithm. Furthermore, he showed that no online algorithm can have a smaller competitive ratio than 2, which is valid even for the case of small jobs. A special case with concave cost was studied by Jiang et al. (2014). For this variant, it was shown that no (non-preemptive or preemptive) algorithm has competitive ratio below 1.5, and algorithms with competitive ratios not exceeding 1.6403 and 1.5654 for the non-preemptive and preemptive variants, respectively, were designed.

In summary, the online problem is defined as follows. We are given a monotonically non-decreasing function $c(m)$. Jobs are presented one by one to the algorithm. For every arriving job, the algorithm can buy additional machines, and then, it is required for it to assign the new job to be processed by its machines in a valid way. For a schedule where the algorithm has m machines and makespan T, its cost is $c(m)+T$. The goal is to minimize the objective function value.

Our contribution We present two new online algorithms for the problem of scheduling with general machine cost function. Both algorithms have smaller competitive ratios than the one known in the literature (Imreh 2009), where these ratios are 2.5 and 2, and their definitions are similar. The algorithms are based on the number of machines that an optimal offline solution uses at every given moment in the sense that the algorithm tries to imitate such a solution and it buys machines whenever its number of machines is smaller. Given the lower bound of Imreh (2009), our 2-competitive algo-

rithm is the best possible, and therefore, we close the problem with general machine cost with and without preemption. Specifically, the algorithms are non-preemptive, though the better algorithm is compared to optimal offline preemptive solutions, and thus, its analysis will be valid for preemptive algorithms, making it an optimal online algorithm among such algorithms as well. The variant of larger competitive ratio is a non-preemptive algorithm. It is presented in order to motivate the usage of the preemptive optimal offline solution for a non-preemptive algorithm, since we show that by replacing the preemptive optimal offline solution with the non-preemptive optimal offline solution, the competitive ratio increased to 2.5.

2 Preliminaries

Throughout the paper, we will use the following notation. The jobs will be labeled $1, 2, \ldots, n$ and presented to the online algorithm in this order. We denote the processing time of job j by $p_j > 0$. The total processing time of the first k jobs is denoted by $P_k = \sum_{j=1}^{k} p_j$, and the maximum processing time of any job in the first k jobs is denoted by $p_k^{\max} = \max_{1 \le j \le k} p_j$. For every input, we will consider a specific optimal offline solution OPT. This offline solution can be optimal among preemptive solutions, or it can be optimal among non-preemptive solutions. We will write which option is analyzed in each case.

Given optimal offline solutions for all prefixes of the input, we will consider the numbers of machines used by these optimal solutions. We will use the term *time j* for the moment in time just after job j arrived and was assigned, and time zero is the time before any job has been presented. Let OPT_j denote the optimal offline solution for time j as well as its cost, where such solutions for distinct values of j can be very different. For an online algorithm A, we let A_j denote the cost for time j. The number of the machines of OPT_j is denoted by m_j^* (where this value is not necessarily monotone in j). For a specific online algorithm (in the case where the algorithm is clear from the context), denote the number of the machines it buys until time j by m_j. Similarly, let T_j^* and T_j denote the values of makespan of OPT_j and of the algorithm at time j, respectively. Recall that the cost of the machines is described by a non-decreasing machine cost function $c(m)$. (This is the total cost of the first m machines.) Thus, $\mathrm{OPT}_j = c(m_j^*) + T_j^*$ and for an algorithm ALG which we analyze, $\mathrm{ALG}_j = c(m_j) + T_j$. We will use $T_0^* = T_0 = 0$ and $m_0^* = m_0 = 0$. (And therefore, $T_1^* > T_0^*$, $T_1 > T_0$, $m_1^* > m_0^*$, and $m_1 > m_0$ will hold.) We use the notation OPT_j^q for the cost of an optimal offline schedule for the prefix of j jobs, under the condition that exactly q machines are used. That is, $\mathrm{OPT}_j = \min_{q=1,2,\ldots,j} \mathrm{OPT}_j^q$, $\mathrm{OPT}_j^{m_j^*} = \mathrm{OPT}_j$, and for any q', we have $\mathrm{OPT}_j^{q'} \ge \mathrm{OPT}_j$ (as the optimal offline solution selects its number of machines optimally). As the input of j jobs contains the input of $j-1$ jobs, the function OPT_j (of costs) is monotonically non-decreasing in j.

We start with a general description of our algorithm. First version of this algorithm runs in exponential time, which is possible for online algorithms. The second version runs in polynomial time.

Algorithm imitate The algorithm acts for every new job as follows. First, compute an optimal offline solution for the input so far, and let m' be the number of machines used by this solution. If the algorithm has less than m' machines, it purchases additional machines such that it will have m' machines. Then, it assigns the new job to the least loaded machine (breaking ties arbitrarily).

Note that when the first job arrives, an optimal offline solution will purchase at least one machine. Therefore, the algorithm will do this as well and it will always have at least one machine. This means that the algorithm will be able to assign the first job (and any other job).

We analyze two variants of Imitate. The algorithm $\mathrm{Imitate}_p$ uses a fixed preemptive optimal offline solution, and the algorithm $\mathrm{Imitate}_n$ uses a fixed non-preemptive optimal offline solution. By the optimal preemptive algorithm (McNaughton 1959), an optimal preemptive solution for m machines has makespan that is the maximum between the largest job and the average load. Thus, for m machines and j jobs, its cost is $c(m) + \max\{\frac{P_j}{m}, p_j^{\max}\}$. Since an optimal offline algorithm can choose any number of machines, its cost is

$$\min_{m=1,2,\ldots,j} \left\{ c(m) + \max\left\{ \frac{P_j}{m}, p_j^{\max} \right\} \right\}.$$

The cost of an optimal offline preemptive solution is computed in polynomial time. We assume that an optimal (offline) non-preemptive solution can be computed as well. This is done, however, in exponential running time (which is allowed for online algorithms).[1]

Obviously, the optimal non-preemptive cost is not smaller than the preemptive one, and if a non-preemptive optimal offline solution uses m machines, its cost is at least $c(m) + \max\left\{\frac{P_j}{m}, p_j^{\max}\right\}$ (but it may be larger). Since it is easier to analyze optimal offline solutions that are chosen by a fixed rule (among optimal solutions), we assume that out of suitable optimal solutions, those with minimum numbers of machines are chosen.

As we use the algorithm LIST SCHEDULING (LS) of Graham (1966), we will use its analysis as follows. After j jobs

[1] We will not discuss how one can use an approximation scheme for converting it into a polynomial time algorithm as our other variant runs in polynomial time and its competitive ratio is smaller.

have been assigned to m machines, the makespan is at most $\frac{P_j}{m} + (1 - \frac{1}{m}) \cdot p_j^{\max} < \frac{P_j}{m} + p_j^{\max}$.

3 Lower bounds

3.1 A general lower bound

We will show later that the competitive ratio of Imitate$_p$ is at most 2. Using the lower bound of Imreh (2009), we will show that this is the exact competitive ratio of the algorithm, and that it is the best possible. Thus, we present a simplified version of the known lower bound that is based on a ski-rental-type approach (Karp 1992). The proof is very similar to that given by Imreh (2009). We provide the proof for completeness and in order to show that the same lower bound holds for preemptive algorithms as well.

Proposition 1 *No preemptive or non-preemptive algorithm for SMC with general costs has competitive ratio below* 2.

Proof For a large integer M, let

$$c(m) = \begin{cases} 0 & \text{if } m = 1 \\ M & \text{if } m \geq 2\,. \end{cases}$$

Consider an arbitrary (preemptive or non-preemptive) online algorithm and a sequence of jobs where each job has processing time 1; $p_j = 1$ for all j. In what follows, we assume that the algorithm may preempt jobs, while an optimal offline solution (also among preemptive solutions) is non-preemptive. Thus, the result holds both for preemptive algorithms and non-preemptive ones.

If the algorithm does not buy a second machine after $2M+2$ jobs have arrived, its makespan is equal to the number of jobs. The competitive ratio is at least 2, as an optimal offline solution can assign one job to each of $2M + 2$ machines, having a makespan of 1 and machine cost M, for a total cost of $M + 1$.

Otherwise, let $\ell \leq 2M + 2$ be the job for which the algorithm buys the second machine. Consider the sequence of ℓ jobs, for which the cost of the algorithm is at least $M + \ell - 1$, as its makespan is $\ell - 1$ before the second machine is bought. First, suppose that $\ell \leq M - 1$. In this case, the competitive ratio of the algorithm is at least $(M + \ell - 1)/\ell \geq (\ell + 1 + \ell - 1)/\ell = 2$, since the algorithm can be compared with an offline algorithm that only purchases one machine, and its cost is ℓ, since this is its makespan.

Now, if $\ell \geq M$, we compare the algorithm to an offline algorithm that purchases ℓ machines and produces a schedule that has makespan 1. Therefore, the optimal offline cost is no larger than $M + 1$. This yields that the competitive ratio is at least $(M + \ell - 1)/(M + 1) \geq (2M - 1)/(M + 1) = 2 - 3/(M + 1)$. As M grows to infinity, this ratio increases to 2. As we considered all possible actions of the algorithm, we found that its competitive ratio is at least 2. □

3.2 A lower bound for Imitate$_n$

Next, we show that the analysis of Imitate$_n$ will not yield a competitive ratio of 2.

Theorem 1 *The competitive ratio of Imitate$_n$ is not smaller than* $\frac{5}{2}$.

Proof Let $k \geq 3$ be a positive integer, and $\varepsilon = \frac{1}{2k}$. Consider an input consisting of two parts as follows. The input is presented as a sequence where the processing times of jobs are stated.

$$I = \underbrace{\underbrace{1, 1, \cdots, 1}_{2k}}_{\text{the first part}}, \underbrace{\underbrace{\varepsilon, \varepsilon, \cdots, \varepsilon}_{2k(2k-4)}, 2}_{\text{the second part}}\,.$$

Thus, the first part of the input consists of $2k$ identical jobs, and the second part of the input is defined by the remaining jobs of sizes ε and 2, respectively. See Figs. 1 and 2 for the two parts of the input.

The costs of machines are as follows. For every i such that $1 \leq i \leq 2k - 1$, the cost is equal to zero, i.e., $c(i) = 0$, the cost of machine $2k$ is $1 - \varepsilon$, i.e., $c(2k) = 1 - \varepsilon$, the cost of machine $2k + 1$ is $999 + \varepsilon$, and for every i such that $i > 2k$, the cost of machine i is zero, so $c(i) = 1000$ for $i \geq 2k + 1$. The machine cost function is therefore defined as follows.

$$c(i) = \begin{cases} 0 & \text{if } 1 \leq i \leq 2k - 1 \\ 1 - \varepsilon & \text{if } i = 2k \\ 1000 & \text{if } i = 2k + 1 \\ 1000 & \text{if } i > 2k + 1 \end{cases}$$

For $j = 1, 2, \ldots, 2k$, we compute (non-preemptive) optimal offline solutions for every prefix of j jobs, and we calculate the value OPT_j for every j, so that we can deduce the numbers of machines used by the algorithm. For $1 \leq j \leq 2k - 1$, it is possible to assign each job to a different machine still paying zero for the machines and obtaining makespan 1. As the jobs have unit sizes, this is optimal, and $\text{OPT}_j = 1$ for $1 \leq j \leq 2k - 1$. Upon arrival of the $(2k)$th job, any solution has two options. It can use at most $2k - 1$ machines, in which case its makespan is at least 2, and the resulting cost is 2, or it can use $2k$ machines (one machine for each job), having makespan 1 and machine cost $1 - \varepsilon$. Thus, $\text{OPT}_{2k} = 2 - \varepsilon$, and $2k$ machines are used by a non-preemptive optimal offline solution. Imitate$_n$ keeps buying

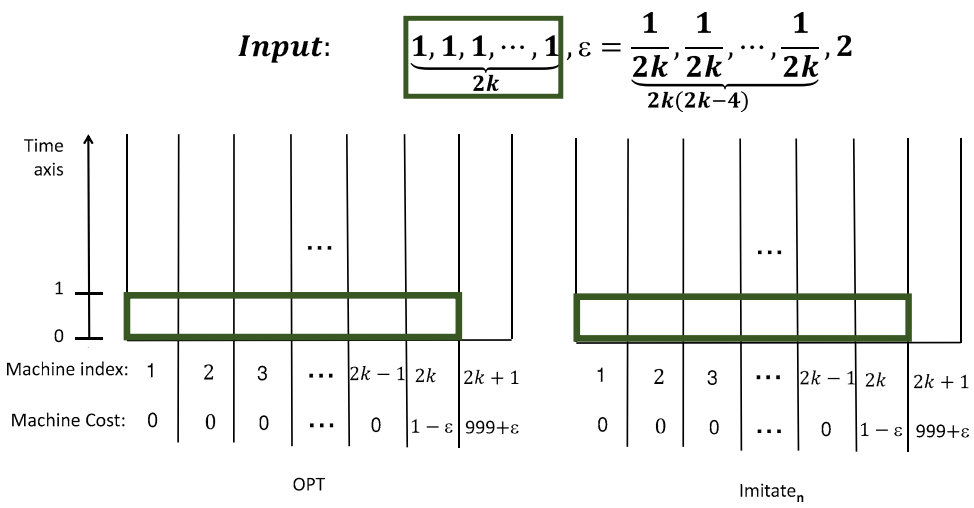

Fig. 1 The input and schedules for the first part of the input discussed in the proof of Theorem 1

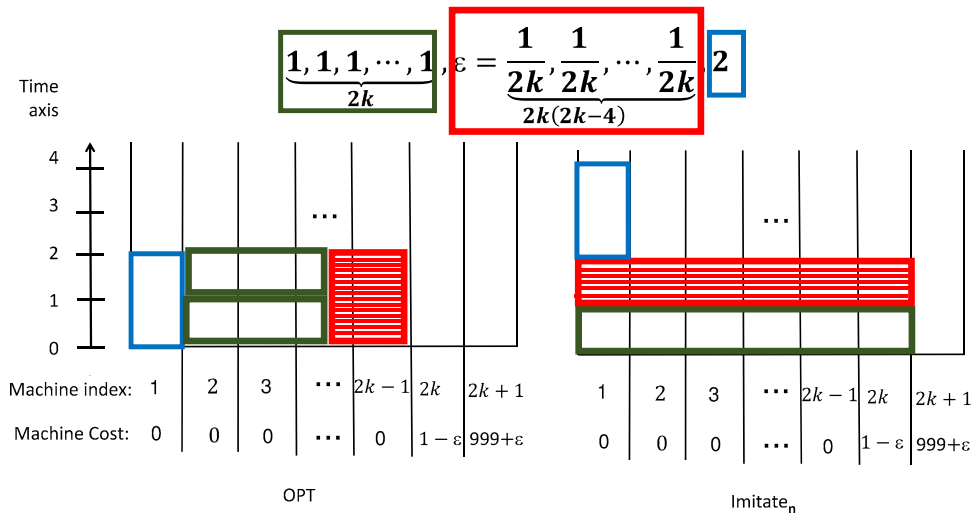

Fig. 2 The input and schedules for the second part of the input discussed in the proof of Theorem 1

one machine for each job, as does the optimal offline solution. At this time, it has $2k$ machines, and one job is assigned to every machine. See Fig. 1 for the schedules.

Now, we consider optimal offline solutions for other prefixes. It is possible to schedule all input jobs on $2k-1$ machines with makespan 2 as follows. One machine has a job of size 2, out of the remaining $2k-2$ machines there are k machines with two unit jobs, and $k-2$ machines with $4k$ jobs out of the other jobs, that is, jobs whose sizes are equal to ε and their total size is 2 for each machine. Thus, as the cost of this solution is 2 (and the cost is not larger if a proper prefix is considered), and none of the optimal offline solutions for the prefixes will buy the machine whose cost is very large (machine $2k+1$). This is an optimal solution for the entire input as there is a job of size 2, so the cost for the entire input cannot be below 2. Thus, for these jobs, Imitate$_n$ will apply LS on $2k$ machines, and it will have $2k-4$ jobs whose sizes are ε assigned to each machine. One machine will also receive the job of size 2, for a makespan of $1+(2k-4)\varepsilon+2 = 4-2/k$. Its cost for the machines is $c(2k) = 1-\varepsilon = 1-1/2k$. See Fig. 2 for the input and the schedules.

Thus, the total cost of the algorithm for the full input is $5-5/2k$. As the cost of an optimal offline solution for the same input is 2, by letting k grow without bound, we observe that the competitive ratio of Imitate$_n$ is at least 2.5. □

4 Algorithms

We start with the analysis of Imitate$_p$.

Theorem 2 *The competitive ratio of Imitate$_p$ is at most 2, no matter whether it is seen as a non-preemptive algorithm or as a preemptive algorithm.*

Proof As the algorithm does not preempt any job, but it is compared to a fixed preemptive optimal offline solution, the

result will hold both for non-preemptive and preemptive algorithms.

To analyze the algorithm, we will focus on two times during its execution for a specific input. Recall that time τ is the time after job τ was assigned.

Given an input of $n \geq 1$ jobs, let $t \leq n$ be the last time when the makespan of the algorithm increases, that is, $T_t > T_{t-1}$ and $T_j = T_t$ for $j \geq t$. Let $s \leq n$ be the last time when the number of machines of the algorithm increases, i.e., the first time that the number of machines of the preemptive optimal offline solution is maximum.

Those times are the time when the algorithm determined its final makespan and final number of machines, respectively. Thus, its final cost for the input is $c(m_n) + T_n = c(m_s) + T_t$.

The main property of our algorithm is that $m_k = \max_{1 \leq j \leq k} m_j^*$. This property holds by the following. Let j be the minimum index for which the right-hand side of this expression is maximized. The algorithm has exactly m_j^* machines starting at time j and until (at least) time k. This implies that the algorithm never purchases additional machines after it purchased at least one machine at time s, and for any $j' \geq s$, it holds that

$$m_{j'}^* \leq m_s^* \,. \tag{1}$$

The case $s \geq t$ In this case, the algorithm assigns the first t jobs to m_t machines. As the makespan of the algorithm increases for job t, all m_t machines had completion times of at least $T_t - p_t$ prior to the assignment of t, so $P_t > P_{t-1} \geq m_t \cdot (T_t - p_t)$ and we get $T_t \leq \frac{P_t}{m_t} + p_t^{\max}$. As $s \geq t$, we have $p_t^{\max} \leq p_s^{\max}$. Moreover, by the definition of the algorithm, $m_t \geq m_t^*$, so $\frac{P_t}{m_t} \leq \frac{P_t}{m_t^*}$. Thus, by $T_s = T_t$, we get $T_s = T_t \leq \frac{P_t}{m_t} + p_t^{\max} \leq \frac{P_t}{m_t^*} + p_s^{\max}$.

The cost of the algorithm satisfies $\mathrm{ALG}_n = \mathrm{ALG}_s = c(m_s^*) + T_s$. Therefore,

$$\mathrm{ALG}_s \leq c(m_s^*) + \frac{P_t}{m_t^*} + p_s^{\max} \,.$$

By

$$\mathrm{OPT}_n \geq \mathrm{OPT}_s = c(m_s^*) + \max\left\{\frac{P_s}{m_s^*}, p_s^{\max}\right\} \geq c(m_s^*) + p_s^{\max}, \tag{2}$$

and by

$$\mathrm{OPT}_s \geq \mathrm{OPT}_t = c(m_t^*) + \max\left\{\frac{P_t}{m_t^*}, p_t^{\max}\right\} \geq \frac{P_t}{m_t^*},$$

we have $\mathrm{ALG}_n \leq 2 \cdot \mathrm{OPT}_n$, as required.

The case $s < t$ First, consider the prefix of the input up to time s. We compare optimal schedules under the condition that the number of machines is fixed to m_s^* or m_t^*. The algorithm never purchases machines after time s, and it has $m_j = m_s$ for $j \geq s$ (so $m_t = m_s$). It also has $m_s = m_s^*$ and $m_j^* \leq m_s$ for $j \geq s$ by (1). We will use the notation defined above: $\mathrm{OPT}_j^m = c(m) + \max\left\{p_j^{\max}, \frac{P_j}{m}\right\}$.

As $\mathrm{OPT}_s = \mathrm{OPT}_s^{m_s^*}$, it holds that $\mathrm{OPT}_s^{m_s^*} \leq \mathrm{OPT}_s^{m_t^*}$, and by writing these costs explicitly, we get

$$c(m_s^*) \leq \max\left\{p_s^{\max}, \frac{P_s}{m_t^*}\right\} - \max\left\{p_s^{\max}, \frac{P_s}{m_s^*}\right\} + c(m_t^*) \,. \tag{3}$$

As in the previous case, we have $T_t \leq \frac{P_t}{m_t} + p_t^{\max} = \frac{P_t}{m_s^*} + p_t^{\max}$, and by (3), we have that

$$\begin{aligned}\mathrm{ALG}_n = \mathrm{ALG}_t = c(m_t) + T_t = c(m_s^*) + T_t \leq c(m_s^*) + \frac{P_t}{m_s^*} + p_t^{\max} \\ \leq \max\left\{p_s^{\max}, \frac{P_s}{m_t^*}\right\} - \max\left\{p_s^{\max}, \frac{P_s}{m_s^*}\right\} + c(m_t^*) + \frac{P_t}{m_s^*} + p_t^{\max} \,.\end{aligned}$$

We consider two sub-cases, based on the value of $\max\{p_s^{\max}, \frac{P_s}{m_t^*}\}$.

If $\max\{p_s^{\max}, \frac{P_s}{m_t^*}\} = p_s^{\max}$, we use $\max\left\{p_s^{\max}, \frac{P_s}{m_s^*}\right\} \geq p_s^{\max}$ and get

$$\mathrm{ALG}_n \leq c(m_t^*) + \frac{P_t}{m_s^*} + p_t^{\max} \leq c(m_t^*) + \frac{P_t}{m_t^*} + p_t^{\max}, \textit{and}$$

by $m_t^* \leq m_s^*$. As $\mathrm{OPT}_t = c(m_t^*) + \max\{\frac{P_t}{m_t^*}, p_t^{\max}\}$, we get $\mathrm{ALG}_n \leq 2 \cdot \mathrm{OPT}_t \leq 2 \cdot \mathrm{OPT}_n$.

If $\max\{p_s^{\max}, \frac{P_s}{m_t^*}\} = \frac{P_s}{m_t^*}$, we use $\max\left\{p_s^{\max}, \frac{P_s}{m_s^*}\right\} \geq \frac{P_s}{m_s^*}$ and have

$$\begin{aligned}\mathrm{ALG}_n &\leq \frac{P_s}{m_t^*} - \frac{P_s}{m_s^*} + c(m_t^*) + \frac{P_t}{m_s^*} + p_t^{\max} \\ &= \frac{P_s}{m_t^*} + c(m_t^*) + \frac{P_t - P_s}{m_s^*} + p_t^{\max}.\end{aligned}$$

As $t > s$, we have $P_t - P_s > 0$, and we have $m_t^* \leq m_s^*$ by (1), so we get

$$\begin{aligned}\mathrm{ALG}_n &\leq \frac{P_s}{m_t^*} + c(m_t^*) + \frac{P_t - P_s}{m_t^*} + p_t^{\max} \\ &= c(m_t^*) + \frac{P_t}{m_t^*} + p_t^{\max} \leq 2 \cdot \mathrm{OPT}_n,\end{aligned}$$

as in the previous sub-case.

Overall, we get $\mathrm{ALG}_n \leq 2 \cdot \mathrm{OPT}_n$ for all cases. □

An analysis of the second algorithm

Next, we analyze Imitate$_n$, to show that the example presented earlier gives a tight bound. This algorithm is a non-preemptive algorithm and analyzed as such. Furthermore, the competitive ratio of Imitate$_n$ is at most 2.5. The following lemma will be used in the proof.

Lemma 1 *Given an input for which an optimal offline solution uses (which is non-preemptive) m machines and has makespan T, it holds that $c(\lceil \frac{m}{2} \rceil) + T \geq c(m)$.*

Proof Given an optimal offline (non-preemptive) solution with m machines, consider an alternative solution as follows. The contents of machines $2i$ and $2i+1$ are assigned to machine i, for $i = 1, \ldots, \lfloor \frac{m}{2} \rfloor$. If m is odd, the contents of machine m are assigned to machine $\frac{m+1}{2}$. The resulting makespan is at most $2T$, and the machine cost is $c(\lceil \frac{m}{2} \rceil)$. The claim follows by optimality of the considered optimal solution. □

Theorem 3 *The competitive ratio of Imitate$_n$ is at most* 2.5.

Proof We follow the structure of the previous proof. Recall that t is the last time when the makespan of the algorithm increases, and s is the last time when the number of machines of the algorithm increases.

The case $s \geq t$ In this case, the proof is very similar. The only part that does not necessarily hold is (2), as it is not always the case, for non-preemptive solutions, that $\mathrm{OPT}_s = c(m_s^*) + \max\left\{\frac{P_s}{m_s^*}, p_s^{\max}\right\}$. However, as m_s^* is the number of machines used by an optimal offline solution, it holds that

$$\mathrm{OPT}_s \geq c(m_s^*) + \max\left\{\frac{P_s}{m_s^*}, p_s^{\max}\right\}.$$

Using this inequality still allows us to continue the proof.

The case $s < t$ In this case, the proof is different, but we still have

$$\mathrm{ALG}_n = \mathrm{ALG}_t = c(m_s^*) + T_t \leq c(m_s^*) + \frac{P_t}{m_s^*} + p_t^{\max}.$$

We split the analysis into the cases $m_s^* \geq 2 \cdot m_t^*$ and $m_s^* \leq 2 \cdot m_t^* - 1$.

In the first case, we have

$$\mathrm{ALG}_t \leq c(m_s^*) + \frac{P_t}{m_s^*} + p_t^{\max} \leq c(m_s^*) + \frac{P_t}{2 \cdot m_t^*} + p_t^{\max},$$

by the condition of the case. As $\mathrm{OPT}_t \geq p_t^{\max}$, $\mathrm{OPT}_t \geq \frac{P_t}{m_t^*}$, and $\mathrm{OPT}_t \geq \mathrm{OPT}_s \geq c(m_s^*)$, we get $\mathrm{ALG}_t \leq 2.5 \cdot \mathrm{OPT}_t$. In this case, the result follows since by decreasing the number of machines significantly, an optimal offline solution increases its makespan.

For the case $m_s^* \leq 2 \cdot m_t^* - 1$, the result will follow as the case where decreasing the number of machines is beneficial implies a certain relation between the machine cost and makespan. The proof of this case is based on Lemma 1.

We apply the lemma with the input consisting of the first s jobs and $m = m_s^*$. We get that

$$c(\lceil \frac{m_s^*}{2} \rceil) + T_s^* \geq c(m_s^*).$$

As in this case $m_t^* \geq \frac{m_s^*+1}{2} \geq \lceil \frac{m_s^*}{2} \rceil$, we have $c(m_t^*) \geq c(\lceil \frac{m_s^*}{2} \rceil) \geq c(m_s^*) - T_s^*$. Therefore, by $c(m_s^*) \leq c(m_t^*) + T_s^*$, we get

$$c(m_s^*) \leq (c(m_s^*) + c(m_t^*) + T_s^*)/2 = (c(m_t^*) + \mathrm{OPT}_s)/2.$$

Thus,

$$\begin{aligned}\mathrm{ALG}_t &\leq c(m_s^*) + \frac{P_t}{m_s^*} + p_t^{\max} \\ &\leq \mathrm{OPT}_s/2 + c(m_t^*)/2 + \frac{P_t}{m_s^*} + p_t^{\max}.\end{aligned}$$

As $\mathrm{OPT}_t \geq p_t^{\max} + c(m_t^*)$, $\mathrm{OPT}_t \geq \frac{P_t}{m_t^*} \geq \frac{P_t}{m_s^*}$ by (1), we get $\mathrm{ALG}_t \leq 2.5 \cdot \mathrm{OPT}_t$. □

References

Chen, B., van Vliet, A., & Woeginger, G. J. (1995). An optimal algorithm for preemptive on-line scheduling. *Operations Research Letters*, *18*(3), 127–131.

Dósa, Gy, & He, Y. (2004). Better online algorithms for scheduling with machine cost. *SIAM Journal on Computing*, *33*(5), 1035–1051.

Dósa, Gy, & He, Y. (2006). Scheduling with machine cost and rejection. *Journal of Combinatorial Optimization*, *12*(4), 337–350.

Dósa, Gy, & Imreh, Cs. (2013). The generalization of scheduling with machine cost. *Theoretical Computer Science*, *510*, 102–110.

Dósa, Gy, & Tan, Z. (2010). New upper and lower bounds for online scheduling with machine cost. *Discrete Optimization*, *7*(3), 125–135.

Fleischer, R., & Wahl, M. (2000). Online scheduling revisited. *Journal of Scheduling*, *3*(6), 343–353.

Graham, R. L. (1966). Bounds for certain multiprocessing anomalies. *Bell System Technical Journal*, *45*(9), 1563–1581.

He, Y., & Cai, S. Y. (2002). Semi-online scheduling with machine cost. *Journal of Computer Science and Technology*, *17*(6), 781–787.

Hochbaum, D. S., & Shmoys, D. B. (1987). Using dual approximation algorithms for scheduling problems: Theoretical and practical results. *Journal of the ACM*, *34*(1), 144–162.

Imreh, Cs. (2009). On-line scheduling with general machine cost functions. *Discrete Applied Mathematics*, *157*(9), 2070–2077.

Imreh, Cs. & Noga, J. (1999). Scheduling with machine cost. In *Proceedings of the 2nd international workshop on approximation algorithms for combinatorial optimization problems (APPROX'99)* (pp. 168–176)

Jiang, Y., & He, Y. (2006). Semi-online algorithms for scheduling with machine cost. *Journal of Computer Science and Technology*, *21*(6), 984–988.

Jiang, Y., & He, Z. (2005). Preemptive online algorithms for scheduling with machine cost. *Acta Informatica*, *41*(6), 315–340.

Jiang, Y., Hu, J., Liu, L., Zhu, Y., & Cheng, T. C. E. (2014). Competitive ratios for preemptive and non-preemptive online scheduling with nondecreasing concave machine cost. *Information Sciences*, *269*, 128–141.

Karp, R. M. (1992) On-line algorithms versus off-line algorithms: How much is it worth to know the future? In *Proceedings of the IFIP 12th world computer congress (IFIP1992), algorithms, software, architecture—information processing, volume A-12 of IFIP transactions* (pp. 416–429).

Leung, J. Y.-T., Lee, K., & Pinedo, M. L. (2012). Bi-criteria scheduling with machine assignment costs. *International Journal of Production Economics*, *139*(1), 321–329.

McNaughton, R. (1959). Scheduling with deadlines and loss functions. *Management Science*, *6*(1), 1–12.

Nagy-György, J., & Imreh, Cs. (2007). On-line scheduling with machine cost and rejection. *Discrete Applied Mathematics*, *155*(18), 2546–2554.

Seiden, S. S., (2000). A guessing game and randomized online algorithms. In *Proceedings of the 32nd annual ACM symposium on theory of computing (STOC'00)* (pp. 592–601).