An $\mathcal{O}(\log m)$-Competitive Algorithm for Online Machine Minimization*

Lin Chen† Nicole Megow† Kevin Schewior‡

October 5, 2015

Abstract

We consider the online machine minimization problem in which jobs with hard deadlines arrive online over time at their release dates. The task is to determine a feasible preemptive schedule on a minimum number of machines. Our main result is a general $\mathcal{O}(\log m)$-competitive algorithm for the online problem, where m is the optimal number of machines used in an offline solution. This is the first improvement on an intriguing problem in nearly two decades. To date, the best known result is a $\mathcal{O}(\log(p_{\max}/p_{\min}))$-competitive algorithm by Phillips et al. (STOC 1997) that depends on the ratio of maximum and minimum job sizes, $p_{\max}$ and $p_{\min}$. Even for $m = 2$ no better algorithm was known. Our algorithm is in this case constant-competitive. When applied to laminar or agreeable instances, our algorithm achieves a competitive ratio of $\mathcal{O}(1)$ even independently of m.

The following two key components lead to our new result. Firstly, we derive a new lower bound on the optimum value that relates the laxity and the number of jobs with intersecting time windows. Then, we design a new algorithm that is tailored to this lower bound and balances the delay of jobs by taking the number of currently running jobs into account.

1 Introduction

Minimizing resource usage is a key to achieving economic, environmental, or societal goals. We consider the fundamental problem of minimizing the number of machines that is necessary for feasibly scheduling preemptive jobs with release dates and hard deadlines. We consider the online variant of this problem in which every job becomes known to the online algorithm only at its release date. We denote this problem as the *online machine minimization problem*. We will show that we may restrict to the *semi-online* problem variant in which the online algorithm is given slightly more information, namely, the optimal number of machines, m, in advance.

In their seminal paper, Phillips, Stein, Torng, and Wein [9] presented a $\mathcal{O}(\log(p_{\max}/p_{\min}))$-competitive algorithm, where $p_{\max}$ and $p_{\min}$ denote the maximum and minimum job processing times. It remained a wide open question if the problem admits a constant-competitive online algorithm [9, 10]. It was not even known whether such an algorithm exists for $m = 2$. Despite serious efforts within the community, no significant improvement has been made within nearly two decades [10].

In this paper we present an $\mathcal{O}(\log m)$-competitive algorithm for the preemptive online machine minimization problem. This is the first result, that depends only on the optimum value, m, instead of other input parameters. Our algorithm is $\mathcal{O}(1)$-competitive when m is bounded or when all jobs have processing time windows which are either agreeable or laminar.

Further related results. The preemptive semi-online machine minimization problem, in which the optimal number of machines is known in advance, has been investigated extensively by Phillips et al. [9]. They show a general lower bound of 5/4 and leave a huge gap to the upper bound $\mathcal{O}(\log(p_{\max}/p_{\min}))$ on the competitive ratio for the so-called *Least Laxity First* (LLF) Algorithm. Not surprisingly, they rule out that the *Earliest Deadline First* (EDF) Algorithm may improve on the performance of LLF by showing a lower bound of $\Omega(\log(p_{\max}/p_{\min}))$.

The non-preemptive variant of our online problem is quite hopeless. In fact, no algorithm can achieve a competitive ratio sublinear in the number of jobs [11]. The non-preemptive problem with unit processing times was studied in a series of papers [4, 6, 7, 11, 12] and implicitly in the context of

*This research was supported by the German Science Foundation (DFG) under contract ME 3825/1. The third author was supported by the DFG within the research training group 'Methods for Discrete Structures' (GRK 1408).

†Technische Universität München, Zentrum für Mathematik, Garching, Germany. Email: {lchen,nmegow}@ma.tum.de.

‡Technische Universität Berlin, Institut für Mathematik, Berlin, Germany. Email: schewior@math.tu-berlin.de.

energy minimization in [1]. It has been shown that an optimal online algorithm has the exact competitive ratio of $e \approx 2.72$ [1, 4].

In a closely related problem variant, an online algorithm is given extra speed to the given number of machines instead of additional unit-speed machines. The goal is to find an algorithm that requires the minimum extra speed. This problem seems much better understood and speedup factors around 2 are known (see [8, 9]). However, the power of speed is much stronger than that of additional machines since it can be viewed to allow parallel processing of jobs to some extent. None of the algorithms that are known to perform well for the speed problem, e.g., EDF and LLF, admit an $f(m)$-competitive algorithm for any function f for the machine minimization problem [9]. Notwithstanding, giving a small amount of extra speed allows for a constant competitive ratio also for the machine minimization problem [8].

We also mention that the offline problem, in which all jobs are known in advance, can be solved optimally in polynomial time if job preemption is allowed [5]. The solution of the natural linear programming (LP) formulation can be rounded in a straightforward way by assigning fractionally assigned workload in a round robin fashion over all machines within a time unit. However, both solutions, the optimum and the LP solutions, may drastically change under online job arrivals.

Our contribution. Our main contribution is a new preemptive online algorithm with a competitive ratio $\mathcal{O}(\log m)$, where m is the optimum number of machines. It is the first improvement upon the longstanding best-known competitive ratio of $\mathcal{O}(\log(p_{\max}/p_{\min}))$ by Phillips et al. [9] – even for $m = 2$. Our algorithm is $\mathcal{O}(1)$-competitive if the optimum value m is bounded. Moreover, the same algorithm is also $\mathcal{O}(1)$-competitive ratio for two important complementary classes of instances, namely laminar and agreeable instances – it achieves the ratio of 96 for laminar instances and 176 for agreeable instances.

We firstly observe that we may restrict to the semi-online model, in which an online algorithm is given the optimum number of machines m in advance. Furthermore, we show that jobs with a small processing time relative to the entire time window ("loose" jobs) are easy to schedule. The difficult task is to schedule "tight" jobs.

The two key components that lead to our new result are a new lower bound on the optimum offline solution m and a new sophisticated delaying scheme for processing jobs. Our lower bound is tailored to "tight" jobs only. While more standard bounds rely on the load in a given interval, we relate for a given set of time intervals the number of jobs with intersecting time windows in these intervals to the fraction that the laxity of those jobs takes of the total interval. The laxity of a job is (informally) the maximum amount of time that a job can be delayed without violating the deadline. For a given optimum value m, our new lower bound construction actually gives an upper bound on the number of jobs with intersecting time intervals, which is how we utilize the result.

To get some intuition for our algorithm, we interpret the laxity of a job as the maximum budget for delaying a job. Whenever a job is not processing (during its feasible time window), its budget is charged by the amount of this delay. Taking this viewpoint, the well-known algorithm Least Laxity First (LLF) is the algorithm that gives priority to the jobs with smallest remaining budgets. Such a naive charging scheme does not give any good bound for general instances [9]. It may seem promising that LLF performs much better when restricting to "tight" jobs. However, this turns out to be not the case as we show LLF does not achieve a competitive ratio of $f(m)$ for any function f even for instances consisting of only "tight" jobs. The reason for failing is that LLF considers only the absolute remaining laxity and ignores the total size of jobs and the amount of time they spent already in the system. In particular, a huge job with huge laxity will be delayed until it is too late while batches of smaller jobs with smaller laxity arrive.

To overcome this problem, we need a more balanced scheme for decreasing the laxity or, equivalently, for using the budget for not-processing a job. Driven by the new lower bound, we design an algorithm that balances the delay of jobs by taking the number of jobs with intersecting intervals into account. We open $m' = \mathcal{O}(m \log m)$ machines and partition the total budget for not-processing a job (i.e., its laxity) into $m' + 1$ "sub-budgets". The i-th "sub-budget" can be accessed only at time points when $i - 1$ jobs are already being processed. We consider the available jobs in decreasing order of release dates and process those with an empty corresponding sub-budget. All other jobs pay for not being processed from their corresponding "sub-budgets". Our main analysis is concerned with relating the algorithm then to the lower bound.

Outline. In Section 2, we define the problem and give general structural insights. We derive a new lower bound on the optimum number of machines in Section 3. The description of our new algorithm is in Section 4 followed by its analysis in Section 5. Finally, in Section 6, we analyze the performance of

our algorithm when being applied to agreeable and to laminar instances, respectively.

2 Problem Definition and Preliminary Results

Problem Definition. Given is a set of n jobs where each job $j \in \{1, 2, \ldots, n\}$ has a processing time $p_j \in \mathbb{N}$, a release date $r_j \in \mathbb{N}$ which is the earliest possible time at which the job can be processed, and a deadline $d_j \in \mathbb{N}$ by which it must be completed. The task is to open a minimum number of machines such that there is a feasible schedule in which no job misses its deadline. In a feasible schedule each job j is scheduled for p_j units of time within the time window $[r_j, d_j)$. Each opened machine can process at most one job at any time, and no job is running on multiple machines at the same time. We allow job preemption, i.e., a job can be preempted at any moment in time and may resume processing later on the same or any other machine.

To evaluate the performance of our online algorithms, we perform a *competitive analysis* (see e.g. [2]). We call an online algorithm *ρ-competitive* if m' machines with $m' \leq \rho \cdot m$ suffice to guarantee a feasible solution for any instance that admits a feasible schedule on m machines.

Notation. For a job j, the *laxity* is defined as $\ell_j = d_j - r_j - p_j$. We call a job *α-loose*, for some $\alpha < 1$, if $p_j \leq \alpha(d_j - r_j)$ and *α-tight* otherwise. The *(processing) interval* of j is $I(j) = [r_j, d_j)$, and j is said to *cover* each $t \in I(j)$. For a set of jobs S, we define $I(S) = \cup_{j \in S} I(j)$. For $I = \cup_{i=1}^{k} [a_i, b_i)$ where $[a_1, b_1), \ldots, [a_k, b_k)$ are pairwise disjoint, we define the *length* of I to be $|I| = \sum_{i=1}^{k} (b_i - a_i)$.

Indexing. Throughout this paper we assume w.l.o.g. that jobs are indexed in increasing order of release dates, and we break ties in decreasing order of deadlines. Hence, for any two jobs j, j' with $j < j'$ one of the three cases holds: (i) $r_j < r_{j'}$, (ii) $r_j = r_{j'}$ and $d_j > d_{j'}$, or (iii) $I(j) = I(j')$.

Characterization of the Optimum. Given an interval I, the *contribution* of a job j to I is $C(j, I) := \max\{0, |I \cap I(j)| - \ell_j\}$, i.e., the minimum processing time that j must receive during I in any feasible schedule. The contribution of a job set S to I is the sum of the individual contributions of jobs in S, and we denote it by $C(S, I)$. Clearly, if S admits a feasible schedule on m machines, $C(S, I)/|I|$ must not exceed m. Interestingly, this bound is tight, which we prove in the full version of the paper.

THEOREM 2.1. *Let m be the minimum number of machines needed to schedule a given job set S feasibly. There exists a union of intervals I with $\lceil C(S, I)/|I| \rceil = m$ but none with $\lceil C(S, I)/|I| \rceil > m$.*

Reduction to the Semi-Online Problem. We show that we may assume that the optimum number of machines m is known in advance by losing at most a factor 4 in the competitive ratio. To do so, we employ the general idea of *doubling* an unknown parameter [3]. More specifically, we open additional machines whenever the optimum solution has doubled.

THEOREM 2.2. *Given a ρ-competitive algorithm for semi-online machine minimization, there is a doubling-based algorithm that is 4ρ-competitive for online machine minimization.*

In the rest of the paper we will thus be concerned with the semi-online problem.

Scheduling α-loose Jobs. We show that, for any fixed $\alpha < 1$, α-loose jobs are easy to handle via a simple greedy algorithm called *Earliest Deadline First* (EDF) on $m' = \rho \cdot m$ machines with $\rho = \mathcal{O}(1)$. This algorithm schedules at any time m' unfinished jobs with the smallest deadline.

THEOREM 2.3. *Let $\alpha \in (0, 1)$. EDF is an $1/(1-\alpha)^2$-competitive algorithm for any instance that consists only of α-loose jobs.*

Scheduling α-tight Jobs. In the remaining part of the paper we assume that all jobs are α-tight for a fixed $\alpha \in (0, 1)$. The good performance of EDF on α-loose jobs, does not apply to instances with α-tight jobs only. In fact, we show the following strong lower bound.

THEOREM 2.4. *For arbitrary $\alpha \in (0, 1)$, EDF has a competitive ratio of $\Omega(n)$ even if $m = 2$ and every job is α-tight.*

The Algorithm *Least Laxity First (*LLF*)* seems a promising candidate for this case. LLF schedules at any time m' unfinished jobs with smallest remaining laxity. However, we show the following negative result.

THEOREM 2.5. *For arbitrary $\alpha \in (0, 1)$, LLF is not $\mathcal{O}(1)$-competitive even if $m = 2$ and every job is α-tight.*

3 A Lower Bound on the Optimum

In this section we derive a new lower bound on m, the offline optimum number of machines. Basic lower bounds for this problem rely on the total load that can be shown to contribute to a (family of) time intervals. To obtain our new bound, we restrict now explicitly to α-tight jobs for some constant $\alpha \in (0,1)$, i.e., $p_j > \alpha(d_j - r_j)$ for any job j. This enables us to relate to the laxity of jobs. The main new ingredient is to take into account the number of intervals covering the time points in a given set of intervals and relate it to the laxity of those jobs. This new lower bound might be of independent interest.

In our setting we are given the optimum m. In that case, the new lower bound allows us to upper bound the number of jobs with intersecting time intervals, which is how we utilize the result.

We use the following definition.

DEFINITION 3.1. *Let G be a set of α-tight jobs and let T be a non-empty finite union of time intervals. For some $\mu \in \mathbb{N}$ and $\beta \in (0,1)$, a pair (G,T) is called* (μ,β)-critical *if*

(i) each $t \in T$ is covered by at least μ distinct jobs in G,

(ii) $|T \cap I(j)| \geq \beta \ell_j$ for any $j \in G$.

In this section, we show the following theorem.

THEOREM 3.1. *If there exists a (μ,β)-critical pair, then $m \geq \frac{\mu-4-4\cdot\lceil\log 8/\beta\rceil}{8+8/\alpha\cdot\lceil\log 8/\beta\rceil} = \Omega\left(\frac{\mu}{\log 1/\beta}\right)$.*

If m is given, this theorem immediately implies the following upper bound on the number of jobs covering the relevant intervals.

COROLLARY 3.1. *If there exists a (μ,β)-critical pair, then $\mu = \mathcal{O}(m \log 1/\beta)$.*

To show the theorem by contradiction, we consider a (μ,β)-critical pair (G,T), and show how to select jobs from G with a total load contribution to some superset of T that contradicts Theorem 2.1.

To that end, the following simple lemma will be useful several times.

LEMMA 3.1. *Let S be a set of jobs. There exists a subset $S' \subseteq S$ such that each time in $I(S)$ is covered by at least one and at most two different jobs in S'.*

A simple greedy algorithm finds such a subset. This result is surely folkloric, but as we could not find a proof, we provide one in the full version of the paper.

Using this simple lemma, we show the first of two important properties of critical pairs.

LEMMA 3.2. *Let (G,T) be a (μ,β)-critical pair. There exist pairwise disjoint subsets $G^\star_1,\dots,G^\star_{\lfloor\mu/4\rfloor}$ of G such that*

(i) $T \subseteq I(G^\star_1) \subseteq \cdots \subseteq I(G^\star_{\lfloor\mu/4\rfloor})$,

(ii) $\sum_{j\in G^\star_i} \ell_j \leq 4|T|/\beta$ for every $1 \leq i \leq \lfloor\mu/4\rfloor$.

Proof. We first select disjoint subsets $G_1,\dots,G_{\lceil\mu/2\rceil}$ of G with a laminar interval structure, i.e., $I(G_1) \subseteq \cdots \subseteq I(G_{\lceil\mu/2\rceil})$, such that each time point in T is covered by at least one and at most two distinct jobs from each G_i: Starting from $i = \lceil\mu/2\rceil$, the iterative procedure selects a job set G_i from $G^+_i := G \setminus (G_{i+1} \cup \cdots \cup G_{\lceil\mu/2\rceil})$ using Lemma 3.1. To also satisfy (ii), we will later further select certain subsets from $G_1,\dots,G_{\lfloor\mu/2\rfloor}$.

Before that, we show that indeed $T \subseteq I(G_1) \subseteq \cdots \subseteq I(G_{\lceil\mu/2\rceil})$. Firstly, we show that $I(G_i) \subseteq I(G_{i+1})$ for every i. This follows from the fact that $I(G^+_1) \subseteq \cdots \subseteq I(G^+_{\lceil\mu/2\rceil})$ (by definition of G^+_i) and $I(G_i) = I(G^+_i)$ for all i (by Lemma 3.1). Secondly, we show $T \subseteq I(G_1)$, i.e., each time point in T is covered by at least one job in G_1. Recall that every time point in T is covered by at least μ distinct jobs in G and, according to Lemma 3.1, covered by at most two distinct jobs in G_i for every i. Hence, any point in T is covered by at least $\mu - 2(\lceil\mu/2\rceil - 1) \geq 1$ jobs in $G^+_1 = G\setminus(G_2\cup\cdots\cup G_{\lceil\mu/2\rceil})$, and $T \subseteq I(G^+_1) = I(G_1)$.

Next we apply a counting argument to show an averaging alternative of Condition (ii). We set $G' = G_1 \cup \cdots \cup G_{\lceil\mu/2\rceil}$, and note that each time in T is covered by at most $2\lceil\mu/2\rceil$ distinct jobs in G' because, as shown above, it is covered by at most two distinct jobs in each G_i. Also using $T \subseteq I(G_1) \subseteq \cdots \subseteq I(G_{\lceil\mu/2\rceil})$, we get

$$|T| = \left|\bigcup_{j\in G'} (T \cap I(j))\right| \geq \frac{\sum_{j\in G'} |T \cap I(j)|}{2\lceil\mu/2\rceil} \geq \frac{\sum_{j\in G'} \beta\ell_j}{2\lceil\mu/2\rceil}, \tag{3.1}$$

where the last inequality follows from the definition of a (μ,β)-critical pair.

Now we argue that we can further choose $G^\star_1,\dots,G^\star_{\lfloor\mu/4\rfloor}$ from the family of job sets $G_1,\dots,G_{\lceil\mu/2\rceil}$ such that Condition (ii) is true. Suppose there are no such sets, i.e., we have $\sum_{j\in G_i} \ell_j > 4\cdot|T|/\beta$ for $\lceil\mu/2\rceil - \lfloor\mu/4\rfloor + 1 \geq \lceil\mu/2\rceil/2$

different i. Then the total laxity of G' is

$$\sum_{j\in G'} \ell_j = \sum_{i=1}^{\lceil \mu/2\rceil} \sum_{j\in G_i} \ell_j$$
$$> \left(\left\lceil \frac{\mu}{2}\right\rceil - \left\lfloor \frac{\mu}{4}\right\rfloor + 1\right)\cdot \frac{4|T|}{\beta} \geq \frac{2\lceil \mu/2\rceil \cdot |T|}{\beta},$$

which is a contradiction to (3.1). □

The following lemma states that, for arbitrary disjoint sets of α-tight jobs with a laminar interval structure, the total size of the covered time intervals is geometrically increasing.

LEMMA 3.3. *Let $S_1, \ldots, S_{\lceil 2m/\alpha\rceil}$ be pairwise disjoint sets of α-tight jobs such that $I(S_1) \subseteq \cdots \subseteq I(S_{\lceil 2m/\alpha\rceil})$. Then we have $|I(S_{\lceil 2m/\alpha\rceil})| \geq 2|I(S_1)|$.*

Proof. Suppose on the contrary that $|I(S_{\lceil 2m/\alpha\rceil})| < 2|I(S_1)|$. We will show a contradiction based on the total contribution of all the jobs to $I(S_{\lceil 2m/\alpha\rceil})$. By assumption we have $|I(S_i)| \geq |I(S_1)| > |I(S_{\lceil 2m/\alpha\rceil})|/2$ for all $i \in \{1, \ldots, \lceil 2m/\alpha\rceil\}$. Each of these sets S_i consists of α-tight jobs only, i.e., $p_j > \alpha(d_j - r_j)$ for any j, and thus, a workload of at least $\alpha \cdot |I(S_i)| \geq \alpha \cdot |I(S_{\lceil 2m/\alpha\rceil})|/2$ has to be processed within the interval $I(S_{\lceil 2m/\alpha\rceil})$. There are $\lceil 2m/\alpha\rceil$ different such sets S_i, and consequently the total workload that has to be processed within $I(S_{\lceil 2m/\alpha\rceil})$ is

$$C\left(\left(\bigcup_{i=1}^{\lceil 2m/\alpha\rceil} S_i\right),\ I(S_{\lceil 2m/\alpha\rceil})\right)$$
$$> \left\lceil \frac{2m}{\alpha}\right\rceil \cdot \frac{\alpha\cdot |I(S_{\lceil 2m/\alpha\rceil})|}{2} \geq m\cdot |I(S_{\lceil 2m/\alpha\rceil})|,$$

which is a contradiction to Theorem 2.1. □

We are now ready to prove the main theorem.

Proof. [Proof of Theorem 3.1.] Let (G,T) be a (μ,β)-critical pair. Let $G^\star_1, \ldots, G^\star_{\lfloor \mu/4\rfloor}$ be subsets of G that satisfy the two conditions of Lemma 3.2, i.e., (i) $T \subseteq I(G^\star_1) \subseteq \cdots \subseteq I(G^\star_{\lfloor \mu/4\rfloor})$, and (ii) $\sum_{j\in G^\star_i} \ell_j \leq 4|T|/\beta$, for all $i \in \{1, \ldots, \lfloor \mu/4\rfloor\}$. The proof idea is to bound the contribution of certain subsets $G^\star_q, \ldots, G^\star_{\lfloor \mu/4\rfloor}$ to the interval $I(G^\star_q)$ and show the bound on m by achieving a contradiction to the load bound in Theorem 2.1.

In the following, we will fix $k \coloneqq \lceil \log 8/\beta\rceil$ and $q \coloneqq \lceil 2m/\alpha\rceil \cdot k$, and we distinguish two cases. If $G^\star_q$ does not exist, i.e., $q > \lfloor \mu/4\rfloor$, we obtain

$$m \geq \frac{\alpha(\mu - 4 - 4k)}{8k} > \frac{\mu - 4 - 4\cdot \lceil \log 8/\beta\rceil}{8 + 8/\alpha \cdot \lceil \log 8/\beta\rceil}$$

as claimed. Otherwise, $G^\star_q$ does exist, and it follows from $T \subseteq I(G^\star_1)$ and repeatedly applying Lemma 3.3 that

$$|I(G_q)| = |I(G^\star_{\lceil 2m/\alpha\rceil \cdot k})| \geq 2^k |T|.$$

Now consider any $G^\star_i$, for $i \in \{1, \ldots, \lfloor \mu/4\rfloor\}$. Using property (ii) of the subsets, we conclude,

$$|I(G^\star_q)| \geq 2^k \cdot |T| \geq 2^k \cdot \frac{\beta}{4} \cdot \sum_{j\in G^\star_i} \ell_j \geq 2\cdot \sum_{j\in G^\star_i} \ell_j, \tag{3.2}$$

where we use $k = \lceil \log 8/\beta\rceil$ in the last step.

For $i \geq q$, the contribution of $G^\star_i$ to $I(G^\star_q)$ can be bounded from below as follows:

$$\begin{aligned} C(G^\star_i, I(G^\star_q)) &= \sum_{j\in G^\star_i} \max\{0, |I(G^\star_q)\cap I(j)| - \ell_j\} \\ &\geq \sum_{j\in G^\star_i} \left(|I(G^\star_q)\cap I(j)| - \ell_j\right) \\ &\geq \left|I(G^\star_i)\cap \bigcup_{j\in G^\star_i} I(j)\right| - \sum_{j\in G^\star_i} \ell_j \\ &= |I(G^\star_i)\cap I(G^\star_q)| - \sum_{j\in G^\star_i} \ell_j. \end{aligned}$$

For $i \geq q$, Equation (3.2) and $I(G^\star_q) \subseteq I(G^\star_i)$ imply $C(G^\star_i, I(G^\star_q)) \geq |I(G^\star_q)|/2$.

We now show that $m > (\lfloor \mu/4\rfloor - q)/2$. Suppose this is not true. Then $\lfloor \mu/4\rfloor \geq q + 2m$, and thus, we have at least $2m+1$ disjoint sets $G^\star_i$ such that $C(G^\star_i, I(G^\star_q)) \geq |I(G^\star_q)|/2$. Hence,

$$C\left(\left(\bigcup_{i=q}^{\mu} G^\star_i\right), I(G^\star_q)\right) \geq (2m+1)\cdot \frac{|I(G^\star_q)|}{2} > m\cdot |I(G^\star_q)|.$$

This is a contradiction to Theorem 2.1. We conclude by noting that indeed

$$m \geq \frac{\mu - 4 - 4\cdot \lceil \log 8/\beta\rceil}{8 + 8/\alpha \cdot \lceil \log 8/\beta\rceil},$$

using $m > (\lfloor \mu/4\rfloor - q)/2$ and our choice of $q = \lceil 2m/\alpha\rceil \cdot k = \lceil 2m/\alpha\rceil \cdot \lceil \log 8/\beta\rceil$. □

We will utilize Theorem 3.1 to construct an $\mathcal{O}(\log m)$-competitive algorithm. To show that this algorithm is $\mathcal{O}(1)$-competitive for the special cases (laminar or agreeable instances), we use a slightly weaker definition of a (μ,β)-critical pair. We replace the job-individual Condition (ii) in Definition 3.1 by an averaging condition.

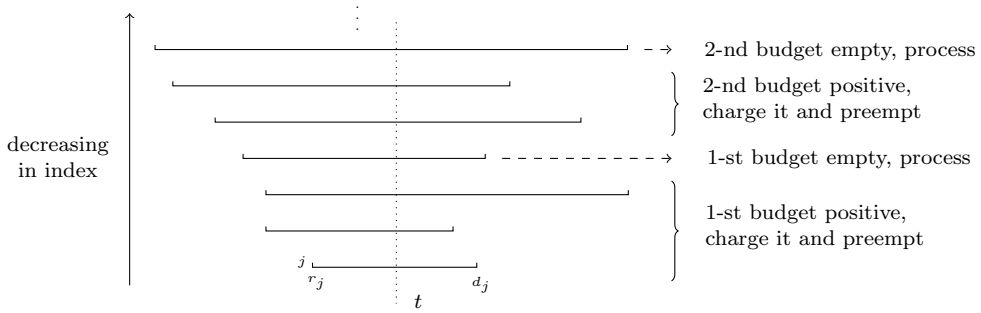

Figure 1: Illustration of our algorithm. At time t, we consider all relevant jobs in reverse order of their indices. After having found i active jobs, we check the $(i+1)$-th budget of the current job. If this budget is not empty, we charge it and preempt/delay the job; otherwise the job becomes active.

DEFINITION 3.2. *Let G be a set of α-tight jobs and let T be a non-empty finite union of time intervals. For some $\mu \in \mathbb{N}$ and $\beta \in (0,1)$, a pair (G,T) is called weakly (μ,β)-critical if*

(i) each $t \in T$ is covered by at least μ distinct jobs in G,

(ii) $|T| \geq \beta/\mu \cdot \sum_{j \in G} \ell_j$.

It is easy to verify that the proofs above apply also to weakly (μ,β)-critical pairs. (Indeed, the only difference is that we do not need the counting argument to obtain Inequality (3.1) in the proof of Lemma 3.2.) Hence, we have the following theorem.

THEOREM 3.2. *If there exists a weakly (μ,β)-critical pair, then $m = \Omega\left(\frac{\mu}{\log 1/\beta}\right)$.*

4 Description of the Algorithm

We assume that the optimum number of machines m is known in advance, and every job is α-tight. We open m' machines and will choose m' later appropriately as a function of m.

The idea for our algorithm is the following. We view the laxity of a job as the budget for delaying it. Whenever a job is delayed for some time, then we pay this delay from its budget. If the budget is empty, we must process the job. Greedily decreasing the budget, as LLF does, fails. Instead, we aim at balancing the decrease of the budget by taking the number of currently processing jobs into account. To that end, we partition the total budget of each job into $m'+1$ equal-size smaller budgets, numbered from 1 to $m'+1$. That is, a job j has $m'+1$ budgets, each of size $\ell_j/(m'+1)$. The i-th budget of a job is reserved for time points when $i-1$ other jobs (with larger index) are being processed, which means that their corresponding 1-st, 2-nd, $\cdots$, $(i-1)$-st budgets have become 0. Once $i-1$ such jobs are already running and the currently considered job has an empty i-th budget, then we process this job even if its other budgets are not empty.

We now describe our algorithm in detail (pseudo-code can be found in the full version of the paper). Figure 1 gives an illustration. At a time t, we consider all jobs with a time window covering t and call them *relevant jobs at t*. We must decide which jobs to process and which jobs to delay/preempt. We call the jobs that are processed at time t *active jobs at time t* and denote them by $a_1(t), a_2(t), \ldots$.

At any computation time t, which we will specify later, we consider all relevant jobs at t in reverse order of their indices, i.e., in decreasing order of release dates. One after the other, we inspect the 1-st budget of each of these jobs. As long as it is positive, we do not process the corresponding job and charge its 1-st budget by the time duration until the next computation time. Once we find a job whose 1-st budget is 0, this job becomes the first active job $a_1(t)$, and each of its budgets remains unchanged until the next computation time. We continue considering jobs in the reverse order of indices, i.e., we consider jobs with index smaller than $a_1(t)$. Now, we inspect the 2-nd budget of jobs (instead of the 1-st budget; because we have one active job). As long as the 2-nd budgets are non-empty we delay the jobs and charge their 2-nd budgets; once we find a job with an empty 2-nd budget, this job becomes the second active job $a_2(t)$. Then we continue with verifying the 3-rd budgets, etc.

The computation times for our algorithm are release dates, deadlines, and time points at which the remaining budget of some job becomes 0. Since the number of budgets per job is $m'+1$, our algorithm has a polynomial running time if the chosen number of machines m' is polynomial.

If we ever find an $(m'+1)$-th active job, we say

that our algorithm *fails*. We will, however, show that we can choose m' such that the algorithm never fails.

5 Analysis of the Algorithm for the General Case

We prove the following theorem.

THEOREM 5.1. *Our algorithm is $\mathcal{O}(\log m)$-competitive.*

By definition of our algorithm, a job's total budget never becomes negative, i.e., it is preempted no longer than its laxity. So we only have to show that our algorithm never finds too many active jobs, i.e., it never finds an $(m'+1)$-th active job. To prove this, we will assume the contrary and will construct a critical pair, from whose existence the theorem then follows by Corollary 3.1.

LEMMA 5.1. *If our algorithm fails, then there exists a $(\mu, 1/\mu)$-critical pair where $\mu = m' + 1$.*

Proof. As our algorithm fails, there is some time $t^\star$ at which an $(m'+1)$-th active job $j^\star$ is found. We construct a $(\mu, 1/\mu)$-critical pair (F, T) which, intuitively speaking, is a minimal subset of jobs still causing the failure. More specifically, we have $F = F_1 \cup \cdots \cup F_\mu$ as well as $T = T_1 \cup \cdots \cup T_\mu$ and, from $i = \mu$ down to $i = 1$, we define F_i as well as T_i as follows. We set $F_\mu := \{j^\star\}$ and $T_\mu := \{t \mid$ the μ-th budget of $j^\star$ is charged at $t\}$. Moreover, for all $i = \mu - 1, \ldots, 1$, we define

$$F_i := \{j \mid j = a_i(t) \text{ for some } t \in T_{i+1} \cup \cdots \cup T_\mu\}$$
$$\text{and } T_i := \{t \mid \text{the } i\text{-th budget of some } j \in F_i \text{ is charged at } t\}.$$

We show that (F, T) is indeed a $(\mu, 1/\mu)$-critical pair. We first show that Condition (i) in Definition 3.1 is satisfied, i.e., each t in T is covered by μ different jobs in F.

By definition of T, for any $t \in T$ there is an i such that $t \in T_i$. Using the definition of T_i, there is some job $j_i \in F_i$ such that its i-th budget is charged at t. Notice that the i-th budget of j_i is charged at time t because there are $i-1$ active jobs $a_1(t), \ldots, a_{i-1}(t)$, all with larger index than j_i and time intervals covering t. Thus, there are at least i jobs j with $j \geq j_i$ that cover t.

We claim that there are at least $\mu - i$ different jobs j with $j < j_i$ that cover t. Assume this is true, then with the claim above, each t in T is indeed covered by μ different jobs in F.

We now prove the claim by (downward) induction on i. Clearly it holds for $i = \mu$. Assume the claim is true for all $h = i+1, \ldots, \mu$. We now show it for i: According to the definition of F_i, j_i is the i-th active job at some $t' \in T_k$ for $k > i$, where $t < t'$ because the remaining i-th budget of j_i at t is still positive, whereas it becomes 0 at t'. By the definition of T_k, there also exists a job $j_k \in F_k$ whose k-th budget is charged at time t'. We apply the induction hypothesis for k to obtain that there are $\mu - k$ different jobs j with $j < j_k < j_i$ covering t'. As $j < j_i$ implies $r_j \leq r_{j_i}$ and we have $t < t'$, all these $\mu - k$ jobs also cover t. Similarly j_k also satisfies $j_k < j_i$ and covers t', so it covers t, too. To finish the proof of the claim, we show that there are $k - i - 1$ different jobs j with $j_k < j < j_i$ that cover t. As the k-th budget of j_k is charged at time t', there must exist active jobs $a_{i+1}(t'), \ldots, a_{k-1}(t')$, each of which covers time t', and we have $j_k < a_h(t') < j_i$ for all $h \in \{i+1, \ldots, k-1\}$. Again using $r_{a_h(t')} \leq r_{j_i} \leq t \leq t'$, all of these $k - i - 1$ jobs cover time t. Hence in total there are $\mu - i$ different jobs j with $j < j_i$ that cover t. We conclude that each t in T is indeed covered by μ different jobs in F.

Finally, we show that also the second property of a $(\mu, 1/\mu)$-critical pair is fulfilled by (F, T). Indeed, $|T \cap I(j)| > \ell_j/\mu$ holds for each $j \in F$: By definition, there is an i such that $j \in F_i$, and thus j is an i-th active job at some time t. As the i-th budget (which initially amounted to ℓ_j/μ) is exhausted at time t, it follows by definition of T_i that $|T_i \cap I(j)| \geq \ell_j/\mu$, and the lemma follows. □

We are now ready to prove the main theorem.

Proof. [Proof of Theorem 5.1.] We show that our algorithm never fails, i.e., it never finds an $(m'+1)$-st active job, when using $m' = \mathcal{O}(m \log m)$ machines. This is sufficient for proving the theorem, as the algorithm opens m' machines and processes at any time the active jobs. All other jobs have a positive corresponding budget and get preempted/delayed. No job is preempted/delayed for more time than its total laxity (budget).

We assume that the algorithm fails, which implies the existence of a $(\mu, 1/\mu)$-critical pair with $\mu = m' + 1$ by Lemma 5.1. Now, Corollary 3.1 implies $\mu \leq cm \log \mu$ for some constant c. Thus, there exists a constant c' such that $m' \leq c' m \log m'$, i.e., the algorithm fails only if $m' \leq c' m \log m'$. For m' sufficiently large, this inequality is not true. Thus, for $m' = \mathcal{O}(m \log m)$ the algorithm does not fail. □

Remark. Careful calculations show that for $m = 2$, our algorithm opens $m' = 236$ machines for tight jobs (taking $\alpha = 4/5$). Including loose jobs (Theorem 2.3) and applying Theorem 2.2 when not

knowning m, our algorithm requires 1044 machines in total, and is thus 522-competitive for the online machine minimization problem when $m = 2$. We remark that we did not optimize the parameters that we choose in the proofs. E.g., in the proof of Theorem 3.1, a more careful analysis could replace $|I(G^\star_{\lceil 2m/\alpha \rceil \cdot k})| \geq 2^k |T|$ by $|I(G^\star_{\lfloor 2m/\alpha \rfloor \cdot k+1})| \geq 2^k |T|$, which already leads to a better ratio of 414 for $m = 2$.

6 Analysis of the Algorithm for Special Cases

In this section we show that our algorithm is constant-competitive for two important special cases, namely, *laminar* and *agreeable* instances. 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'.

THEOREM 6.1. *For laminar and agreeable instances, our algorithm is $\mathcal{O}(1)$-competitive.*

Recall that we only consider α-tight jobs. As in the general case, our proof strategy is to construct a (here: weakly) critical failure pair whenever our algorithm finds a μ-th active job $j^\star$. To prove a constant competitive ratio, however, we use a slightly different construction in which we drop active jobs with intersecting intervals, and obtain a weakly (μ, β)-critical pair (H, T) where $\beta = 1$. This directly implies the theorem by Corollary 3.1[1]. In fact, the construction is identical for both special cases. We construct job set $H \coloneqq H_1 \cup \cdots \cup H_\mu$ and time points $T \coloneqq T_1 \cup \cdots \cup T_\mu$ by (downward) inductively defining $H_\mu \coloneqq \{j^\star\}$, $T_\mu \coloneqq \{t \mid \text{the } \mu\text{-th budget of } j^\star \text{ is charged at } t\}$,

$$F_i \coloneqq \{j \mid j = a_i(t) \text{ for some } t \in T_{i+1} \cup \cdots \cup T_\mu\},$$
$$H_i \coloneqq \{j \in F_i \mid \text{there does not exist } j' \in F_i \text{ s.t. } I(j) \cap I(j') \neq \emptyset \text{ and } j' < j\},$$

and $T_i \coloneqq \{t \mid \text{the } i\text{-th budget of some } j \in H_i \text{ is charged at } t\}$,

for all $i = \mu - 1, \ldots, 1$. Note that the only difference from the construction for the general case is that for each set F_i we additionally maintain a set $H_i \subseteq F_i$ in which we keep for any two intersecting intervals in F_i only one. We call (H, T) the *failure pair*.

To make more concise statements about the structure of the constructed pair, we introduce the notation $S_1 \prec S_2$ (or equivalently, $S_2 \succ S_1$) for two sets of jobs S_1 and S_2. Specifically, this means that for every job $j_2 \in S_2$ there exists a job $j_1 \in S_1$ such that $I(j_1) \cap I(j_2) \neq \emptyset$ and $j_1 < j_2$. Note that $\prec$ is not an order as it does not necessarily obey transitivity.

The following structural lemma is true for both special cases; see the full version of the paper for the individual proofs.

LEMMA 6.1. *Consider a laminar or agreeable instance, and let (H, T) be a failure pair. Then the following statements are true:*

(i) For all $H_i, H_{i'}$ with $i < i'$, we have $H_i \succ H_{i'}$.

(ii) For all H_i, we have $T \subseteq I(H_i)$.

With Lemma 6.1, we can prove Theorem 6.1 without using any further structural information.

Proof. [Proof of Theorem 6.1.] We show that our algorithm never finds a μ-th active job if $\mu - 1 = m' = cm$ for a sufficiently large constant c. To this end, assume the contrary and let (H, T) be the corresponding failure pair, which we claim to be weakly $(\mu, 1)$-critical. Given this claim, the theorem directly follows from Theorem 3.2.

The first property of a weakly $(\mu, 1)$-critical pair, that is, that each $t \in T$ is covered by μ different jobs in H, is easy to see: By Lemma 6.1 (ii), there is a job in each H_i that covers t. Also, all these jobs are distinct: If there exists a job $j \in H_i \cap H_{i'}$ where $i < i'$, Lemma 6.1 (i) implies the existence of $j' \in H_{i'}$ with $j' < j$ and $I(j) \cap I(j') \neq \emptyset$. This would be a contradiction to the construction of (H, T) as j would not be taken over from $F_{i'}$ to $H_{i'}$ because $j' \in H_{i'}$, $I(j) \cap I(j') \neq \emptyset$, and $j' < j$.

As an intermediate step, we claim that $T_1, \ldots, T_\mu$ are pairwise disjoint. To see this, suppose there exists some $t \in T_i \cap T_{i'}$ where $i < i'$. By definition of T_i, there exists a job $j \in H_i$ such that the i-th budget of j is charged at t. Similarly, there also exists some job whose i'-th budget is charged at t, implying that at time t there exists an i-th active job $a_i(t)$ as $i < i'$. We distinguish three cases, each of them yielding a contradiction:

Case 1: We have $a_i(t) < j$. Note that $a_i(t) \in F_i$ by definition of F_i. As $t \in I(j) \cap I(a_i(t)) \neq \emptyset$, we get a contradiction as, by definition of H_i, it does not include j.

Case 2: We have $a_i(t) = j$. Then the i-th budget of j is already exhausted at t, which is a contradiction to the fact that it is charged at t.

[1] We are dropping constants by writing $m = \Omega(\mu / \log(1/\beta))$ in Theorem 3.2. The logarithm is actually taken over $8/\beta$ instead of $1/\beta$ (see Equation (3.2)), and thus, $\beta = 1$ does not cause a problem in computation.

Case 3: We have $a_i(t) > j$. Recall that the algorithm considers jobs one by one in decreasing order of indices. After it found an i-th active job $a_i(t)$ at t, it will never check the i-th budget of jobs of lower indices. Hence the i-th budget of job $j < a_i(t)$ cannot be charged at t.

It remains to show the second property of a weakly $(\mu, 1)$-critical pair, i.e., we have $|T| \geq \sum_{j \in F} \ell_j/\mu$. For each H_i, every $j \in H_i$ is an i-th active job at some time t and thus its i-th budget is exhausted at t. Consequently, there are times at which this budget is charged, implying $|T_i \cap I(j)| \geq \ell_j/\mu$. Using that H_i does not contain distinct jobs j and j' with $I(j) \cap I(j') \neq \emptyset$ (by definition), we obtain

$$|T_i| = \sum_{j \in H_i} |T_i \cap I(j)| \geq \sum_{j \in H_i} \frac{\ell_j}{\mu}.$$

As $T_1, \ldots, T_\mu$ are pairwise disjoint, by summing up these inequalities for all H_i, we get:

$$|T| = \sum_{i=1}^{\mu} |T_i| \geq \sum_{i=1}^{\mu} \sum_{j \in H_i} \frac{\ell_j}{\mu} = \sum_{j \in H} \frac{\ell_j}{\mu},$$

which concludes the proof. □

Remark. Careful calculations show that, when restricting to only α-tight jobs, the algorithm is $(\lceil 8/\alpha \rceil + 4)$-competitive for laminar instances, and $(\lceil 16/\alpha \rceil + 8)$-competitive for agreeable instances. Note that the factor of two between the ratios is due to that fact that for laminar instances the H_i's we derive already satisfies a laminar structure, i.e., $I(H_1) \subseteq I(H_2) \subseteq \cdots \subseteq I(H_\mu)$, while for agreeable instances we still need to apply Lemma 3.1 to get such a structure. When additionally handling loose jobs by EDF (Theorem 2.3), we derive a 24-competitive algorithm for laminar instances and a 44-competitive algorithm for agreeable instances if the offline optimum m is known (by taking $\alpha = 1/2$). If m is not known, using Theorem 2.2 we derive a 96-competitive algorithm for laminar instances and a 176-competitive algorithm for agreeable instances.

7 Conclusions

We presented an improved online algorithm for the preemptive machine minimization problem. Our general $\mathcal{O}(\log m)$-competitive algorithm yields a competitive ratio of $\mathcal{O}(1)$ for several special cases, such as, laminar and agreeable instances as well as instances where the optimum value m is bounded by a constant.

Our methodology is based on a new lower bound construction which may be of independent interest. We cannot rule out that a different construction of a failure set in the analysis may give a (μ, β)-critical pair with a constant β, which would prove directly a constant competitive ratio. Indeed, we show how to achieve this for laminar and agreeable instances. The results on these special cases verify the applicability of our general method, but we mention that algorithm and analysis are not optimized to give best possible constants.

References

[1] N. Bansal, T. Kimbrel, and K. Pruhs. Speed scaling to manage energy and temperature. *J. ACM*, 54(1), 2007.

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

[3] M. Chrobak and C. Kenyon-Mathieu. SIGACT news online algorithms Column 10: Competitiveness via doubling. *SIGACT News*, 37(4):115–126, 2006.

[4] N. R. Devanur, K. Makarychev, D. Panigrahi, and G. Yaroslavtsev. Online algorithms for machine minimization. *CoRR*, abs/1403.0486, 2014.

[5] W. A. Horn. Some simple scheduling algorithms. *Naval Research Logistics Quarterly*, 21(1):177–185, 1974.

[6] M.-J. Kao, J.-J. Chen, I. Rutter, and D. Wagner. Competitive design and analysis for machine-minimizing job scheduling problem. In *Proc. of ISAAC*, pages 75–84, 2012.

[7] A. J. Kleywegt, V. S. Nori, M. W. P. Savelsbergh, and C. A. Tovey. Online resource minimization. In *Proc. of SODA*, pages 576–585, 1999.

[8] T. W. Lam and K.-K. To. Trade-offs between speed and processor in hard-deadline scheduling. In *Proc. of SODA*, pages 623–632, 1999.

[9] C. A. Phillips, C. Stein, E. Torng, and J. Wein. Optimal time-critical scheduling via resource augmentation. In *Proc. of STOC*, pages 140–149, 1997.

[10] K. Pruhs. In *Collection of Open Problems in Scheduling*, Dagstuhl Scheduling Seminar, 2010.

[11] B. Saha. Renting a cloud. In *Proc. of FSTTCS*, pages 437–448, 2013.

[12] Y. Shi and D. Ye. Online bin packing with arbitrary release times. *Theor. Comput. Sci.*, 390(1):110–119, 2008.