SIAM J. COMPUT.
Vol. 47, No. 6, pp. 2057–2077

AN $\mathcal{O}(\log m)$-COMPETITIVE ALGORITHM FOR ONLINE MACHINE MINIMIZATION*

LIN CHEN†, NICOLE MEGOW‡, AND KEVIN SCHEWIOR§

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 to 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. [*Optimal time-critical scheduling via resource augmentation*, 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. First, 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.

Key words. online algorithms, multiprocessor scheduling, competitive analysis

AMS subject classifications. 68Q25, 68W27, 68W40, 90B35

DOI. 10.1137/17M116032X

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 jobs with release dates and hard deadlines. Jobs may be preempted and migrated between machines. 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 et al. [18] presented an algorithm with competitive ratio $\mathcal{O}(\log(p_{\max}/p_{\min}))$, where $p_{\max}$ and $p_{\min}$ denote the maximum and minimum job processing times. It remained a wide open question whether the problem admits a constant-competitive online algorithm [18, 20]. 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 in nearly two decades [20].

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

*Received by the editors December 7, 2017; accepted for publication (in revised form) August 22, 2018; published electronically November 27, 2018. An extended abstract of this work appeared in ACM-SIAM Symposium on Discrete Algorithms (SODA), 2016, pp. 155–163.

http://www.siam.org/journals/sicomp/47-6/M116032.html

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

†University of Houston, Houston, TX 77004 (chenlin198662@gmail.com).

‡Universität Bremen, Bremen, Germany (nicole.megow@uni-bremen.de).

§Technische Universität München, Munich, Germany, and École Normale Supérieure Paris, PSL University, Paris, France (kschewior@gmail.com).

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.

1.1. 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. [18]. They showed a general lower bound of $5/4$ and left a huge gap on the upper bound $\mathcal{O}(\log(p_{\max}/p_{\min}))$ of the competitive ratio for the so-called least laxity first (LLF) algorithm. Not surprisingly, they ruled out that the earliest deadline first (EDF) algorithm may improve on the performance of LLF by showing a lower bound of $\Omega(p_{\max}/p_{\min})$.

The nonpreemptive variant of our online problem is quite hopeless. In fact, no algorithm can achieve a competitive ratio sublinear in the number of jobs [21]. The nonpreemptive problem with unit processing times was studied in a series of papers [9, 15, 16, 21, 22] and implicitly in the context of energy minimization in [2]. It has been shown that an optimal online algorithm has the exact competitive ratio of $e \approx 2.72$ [2, 9].

There is also a strong lower bound when preemption is allowed but there is no migration between machines: in this case, no algorithm can achieve a competitive ratio sublogarithmic in the number of jobs [5]. Nevertheless, there is an $\mathcal{O}(1)$-competitive algorithm for agreeable instances and the nonpreemptive problem, and there is an $\mathcal{O}(\log m)$-competitive algorithm for laminar instances and the nonmigratory problem [5].

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 [17, 18]). However, the power of speed is much stronger than that of additional machines since it can be seen 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 our problem [18]. In addition, when migration is not allowed, there is an algorithm requiring only constant speed [4] but no algorithm is $f(m)$-competitive for any function f for our problem [5]. Notwithstanding, giving any constant extra speed allows for a constant competitive ratio also for the machine minimization problem [17].

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 [12]. 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.

Again, the problem complexity increases drastically if preemption is not allowed. In fact, the problem of deciding whether one machine suffices to schedule all the jobs nonpreemptively is strongly NP-complete [11]. It is even open whether a constant-factor approximation exists; a lower bound of $2-\varepsilon$ was given in [8]. The currently best known nonpreemptive algorithm has an approximation factor of $\mathcal{O}(\sqrt{(\log n)/(\log\log n)})$ [7]. Small constant factors were obtained for special cases [8, 23]. However, when the speed of the machines is slightly increased, then the general problem can also be approximated within a factor of 2 [13].

1.2. 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 up the longstanding best-known competitive ratio of $\mathcal{O}(\log(p_{\max}/p_{\min}))$ by Phillips et al. [18]—even for $m = 2$. Our algorithm is $\mathcal{O}(1)$-competitive if the optimum value m is bounded. Moreover, the same algorithm has an $\mathcal{O}(1)$-competitive ratio for two important complementary classes of instances, namely laminar and agreeable instances—it achieves a ratio of 96 for laminar instances and 176 for agreeable instances.

We first observe that we may restrict ourselves 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 LLF is the algorithm that gives priority to the jobs with the smallest remaining budgets. Such a naive charging scheme does not give any good bound for general instances [18]. It may seem promising that LLF performs much better when restricted 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 have 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 delays 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 ith "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 date 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.

Very recently, our result was improved to an $\mathcal{O}(\log m/\log\log m)$-competitive algorithm [1] and subsequently to an $\mathcal{O}(\log\log m)$-competitive algorithm [14]. The former result was achieved by optimizing the parameter according to which the jobs are classified as loose or tight. The latter paper introduced $O(\log\log m)$ classes of jobs

between loose and tight jobs, all of which are handled on separate sets of machines by preferring (in an absolute sense) short jobs. Both papers crucially build on our algorithm and our analysis, or a parameterized version thereof.

1.3. Outline. In section 2, we define the problem and give general structural insights, allowing us to restrict ourselves to the semi-online problem of tight jobs. 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 applied to agreeable and to laminar instances, respectively.

2. Problem definition and preliminary results.

2.1. Problem definition. Given are a set of jobs where each job $j \in \{1, 2, \dots, n\}$ 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 on any other machine.

To evaluate the performance of our online algorithms, we perform a *competitive analysis* (see, e.g., [3]). We call an online algorithm *ρ-competitive* if it requires no more than $\rho \cdot m$ machines to guarantee a feasible solution for any instance that admits a feasible schedule on m machines.

2.2. Notation and indexing. For all job parameters (such as release dates, processing times, etc.) x_j, we set $x_{\min} \coloneqq \min_j x_j$ and $x_{\max} \coloneqq \max_j x_j$, taken over the entire instance. 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 $p_j \le \alpha(d_j - r_j)$ 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 $[a_1, b_1), \dots, [a_k, b_k)$ are pairwise disjoint, we define the *length* of I to be $|I| = \sum_{i=1}^{k} (b_i - a_i)$.

Throughout this paper we assume without loss of generality (w.l.o.g.) that jobs are indexed in increasing order of release date, and we break ties in decreasing order of deadline. Hence, for any two jobs j, j' with $j < j'$ one of these three cases holds: (i) $r_j < r_{j'}$, (ii) $r_j = r_{j'}$ and $d_j > d_{j'}$, or (iii) $I(j) = I(j')$.

2.3. Characterization of the optimum. Given a finite union of intervals I, the *contribution* of a job j to I is $C(j, I) \coloneqq \max\{|I \cap I(j)| - \ell_j, 0\}$, 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.

THEOREM 1. *Let S be a set of jobs. Then S can be feasibly scheduled on m machines if and only if $C(S, I)/|I| \le m$ for all finite unions of intervals I.*

In the proof, we will make use of the following construction due to Horn [12], which reduces the problem of deciding the existence of a feasible schedule of S on m machines to a maximum single-commodity flow problem: We construct a flow network

$N_{S,m}$ consisting of a directed graph $G_{S,m} = (V, A)$, a source-sink pair $s, t \in V$, and a capacity function $c : A \to \mathbb{N}$. We define $V \coloneqq \{s, t\} \cup V_S \cup V_T$, where

$$V_S \coloneqq \{u_j \mid j \in S\} \text{ and } V_T \coloneqq \{v_\vartheta \mid \vartheta \in \mathbb{N}, r_{\min} \leq \vartheta < d_{\max}\}$$

correspond to the job set S and the set of relevant integer time points, respectively. For each job $j \in S$, we add unit-capacity edges $(u_j, v_{r_j}), \ldots, (u_j, v_{d_j-1})$. Furthermore, for each job j we add an edge (s, u_j) with capacity p_j, and for each time ϑ we add an edge (v_ϑ, t) with capacity m.

We first restate the following result, which was originally proved by Horn [12] using an LP which implicitly solves the flow problem considered here.

PROPOSITION 2 (Horn [12]). *There exists an s-t flow of value $\sum_{j \in S} p_j$ in $N_{S,m}$ if and only if there is a feasible schedule of S on m machines.*

Using this property together with the max-flow-min-cut theorem [10] now essentially shows the theorem.

Proof of Theorem 1. It is obvious that, if there is a feasible schedule of S on m machines, $C(S, I) \leq m \cdot |I|$ for all finite union of intervals I: If there is a feasible schedule of S on m machines, which schedules a total volume of at most $m \cdot |I|$ within I, the contribution of S to I is clearly at most $m \cdot |I|$.

It remains to show that, if S cannot be scheduled feasibly on m machines, we have $C(S, I^\star) > m \cdot |I^\star|$ for some finite union of intervals $I^\star$. By the assumption, applying Proposition 2 and the max-flow-min-cut theorem yields that the minimum s-t cut $C \subsetneq V$ in $N_{S,m}$ is of value less than $\sum_{j \in S} p_j$. By writing the cut value more explicitly, we get

$$\sum_{j : u_j \in V_S \setminus C} p_j + \sum_{j : u_j \in V_S \cap C} |N^+(v_j) \setminus C| + \sum_{\vartheta : v_\vartheta \in V_T \cap C} m < \sum_{j \in S} p_j, \tag{1}$$

where $N^+(v)$ denotes the set of vertices reachable from vertex v in one step. Equivalently, we have

$$\sum_{j : u_j \in V_S \cap C} p_j - \sum_{j : u_j \in V_S \cap C} |N^+(v_j) \setminus C| > \sum_{\vartheta : v_\vartheta \in V_T \cap C} m.$$

If we define $I^\star \coloneqq \bigcup_{\vartheta : v_\vartheta \in V_T \cap C} [\vartheta, \vartheta + 1)$, we thus get

$$\sum_{j \in S} \max\{p_j - |I(j) \setminus I^\star|, 0\} \geq \sum_{j : u_j \in V_S \cap C} p_j - |I(j) \setminus I^\star| > m \cdot |I^\star|.$$

Since the left-hand side is exactly $C(S, I^\star)$, the claim follows. ∎

2.4. 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 of 4 in the competitive ratio. To do so, we employ the general idea of *doubling* an unknown parameter [6]. More specifically, we open additional machines whenever the optimum solution has doubled.

THEOREM 3. *Given a ρ-competitive algorithm for semi-online machine minimization, there is a doubling-based algorithm that is 4ρ-competitive for online machine minimization.*

Towards describing the algorithm, denote by $\mathsf{A}(m')$ the semi-online algorithm A that is given m' as number of machines. Further, denote by $m(t)$ the minimum number of machines needed to feasibly schedule all jobs released up to time t. Then our algorithm for the fully online problem is as follows.

Algorithm A': Let $t_0 \coloneqq \min_j r_j$. For $i = 1, 2, \ldots$, if $t_{i-1} \neq \infty$,
- let $t_i \coloneqq \min\{t \mid m(t) > 2m(t_{i-1})\} \cup \{\infty\}$;
- all jobs released within $\in [t_{i-1}, t_i)$ are run by Algorithm $\mathsf{A}(2m(t_{i-1}))$ on a separate set of machines.

Since the time points $t_0, t_1, \ldots$ as well as $m(t_0), m(t_1), \ldots$ can be computed online and A is assumed to be an algorithm for the semi-online problem, this procedure can be executed online. We prove that this algorithm indeed only requires $4\rho m$ machines to produce a feasible schedule.

Proof of Theorem 3. For any i such that $t_i \neq \infty$, first note that the set S_i of all jobs released within $[t_0, t_{i+1})$ has a feasible schedule on $m(t_{i+1}-1) \leq 2m(t_i)$ machines. Thus, as a subset of S_i, also the set S_i' of all jobs released within $r_j \in [t_i, t_{i+1})$ has a feasible schedule on $2m(t_i)$ machines. Consequently, by definition of A, $\mathsf{A}(2m(t_i))$ produces a feasible schedule S_i' on $2\rho m(t_i)$ machines.

It remains to compare the number of machines opened by A' with the minimum number of machines m needed to schedule the whole input instance. Let k be maximal such that t_k is defined. Then the total number of machines opened by A' is

$$\sum_{i=0}^{k} 2\rho m(t_i) \leq \sum_{i=0}^{k} \frac{2\rho}{2^{k-i}} \cdot m(t_k) = 2\rho \sum_{i=0}^{k} \frac{1}{2^i} \cdot m(t_k) < 4\rho m,$$

where we use $2m(t_i) \leq m(t_{i+1})$ for all $i \in \{0, \ldots, k-1\}$ in the first step and $m(t_k) \leq m$ in the last step. This concludes the proof. ∎

In the rest of the paper we focus on the semi-online problem.

2.5. 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 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 (or all, if less are available). If there are multiple jobs with the same deadline, we assume that EDF breaks the tie arbitrarily but consistently over time.

THEOREM 4. *Let $\alpha \in (0, 1)$. EDF is an $1/(1-\alpha)^2$-competitive algorithm for any instance that consists only of α-loose jobs.*

Towards the proof of this theorem, recall that EDF on m' machines is a busy algorithm; that is, whenever there are at most m' unfinished jobs available, EDF schedules all of them. Similar to the analysis of EDF in the context of speed augmentation given in [19], we obtain the following helpful lemma for all busy algorithms. Here, we denote by $W^{\mathsf{A}}(t)$ and $W^{\mathsf{OPT}}(t)$ the total workload (released or unreleased) that is remaining for algorithm A and an arbitrary optimum OPT.

LEMMA 5. *Consider $\alpha \in (0, 1)$, instances consisting of α-loose jobs, and a busy online algorithm A using $m/(1-\alpha)^2$ machines making decisions only at integer time points. For all $t \leq d_{\max}$, we have*

$$W^{\mathsf{A}}(t) \leq W^{\mathsf{OPT}}(t) + \frac{\alpha}{1-\alpha} \cdot m \cdot (d_{\max} - t)\,.$$

Proof. We prove by induction on integer time points. The base is clear because $W^A(0) = W^{\mathsf{OPT}}(0)$. Now we assume the statement holds for all $t' < t$ and show it for t. We consider three cases. Note that at least one case occurs, and possibly more than one occurs.

Case 1: A processes at least $m/(1-\alpha)$ jobs at time t. According to the induction hypothesis, it holds that

$$W^{\mathsf{A}}(t-1) \leq W^{\mathsf{OPT}}(t-1) + \frac{\alpha}{1-\alpha} \cdot m \cdot (d_{\max} - t + 1). \tag{2}$$

Since the optimum can at most finish a workload of m within the time interval $[t, t+1)$, we get $W^{\mathsf{OPT}}(t) \geq W^{\mathsf{OPT}}(t-1) - m$. For Algorithm A, it holds that $W^{\mathsf{A}}(t) \leq W^{\mathsf{A}}(t-1) - m/(1-\alpha)$. Inserting the two latter inequalities into inequality 2 proves the claim.

Case 2: A processes less than $m/(1-\alpha)$ jobs at time t, and we have $p_j(t) \leq \alpha(d_j - t)$ for all unfinished jobs. Again since A is busy and there is an empty machine at time t, there are less than $m/(1-\alpha)$ unfinished jobs. Then the total remaining processing time of unfinished jobs is bounded by $\alpha \cdot (d_{\max} - t) \cdot m/(1-\alpha)$. Plugging in the total processing time of unreleased jobs, which is bounded by $W^{\mathsf{OPT}}(t)$, we again get the claim.

Case 3: There exists an unfinished job j with $p_j(t) > \alpha(d_j - t)$. Using that j is α-loose, i.e., $p_j \leq \alpha(d_j - r_j)$, we get that j has not been processed for at least $(1-\alpha)(t-r_j)$ time units in the interval $[r_j, t)$. As Algorithm A is busy, this means that all machines are occupied at these times, yielding

$$\begin{aligned} W^{\mathsf{A}}(t) &\leq W^{\mathsf{A}}(r_j) - (1-\alpha)(t-r_j) \cdot \frac{m}{(1-\alpha)^2} \\ &\leq W^{\mathsf{OPT}}(r_j) + \frac{\alpha}{1-\alpha} \cdot m \cdot (d_{\max} - r_j) - (1-\alpha)(t-r_j) \cdot \frac{m}{(1-\alpha)^2} \\ &\leq W^{\mathsf{OPT}}(r_j) + \frac{\alpha}{1-\alpha} \cdot m \cdot (d_{\max} - t) - m \cdot (t - r_j), \end{aligned}$$

where the second inequality follows by the induction hypothesis for $t = r_j$, and the third one follows from elementary transformations. Last, the feasibility of the optimal schedule implies

$$W^{\mathsf{OPT}}(t) \geq W^{\mathsf{OPT}}(r_j) - m \cdot (t - r_j),$$

which in turn implies the claim by plugging it into the former inequality. This completes the proof. ∎

Proving the theorem is now simple.

Proof of Theorem 4. Suppose that EDF using $m/(1-\alpha)^2$ machines fails on a feasible instance J. Out of the jobs that EDF fails to schedule, let $j^\star$ be the job with the earliest deadline. Let $J^\star := \{j \mid d_j \leq d_{j^\star}\}$ and note that EDF's processing of $J^\star$ is unaffected of the presence of $J \setminus J^\star$. Hence EDF on $J^\star$ still fails to meet the deadline of $j^\star$, allowing us to consider $J^\star$ instead of J from now on. Now we apply Lemma 5, showing $W^{\mathsf{A}}(d_{j^\star}) \leq W^{\mathsf{OPT}}(d_{j^\star}) = 0$, meaning that A has finished *all* of the workload at time $d_{j^\star}$, and in particular it has finished $j^\star$. Hence, A does not miss the deadline of $j^\star$; a contradiction. ∎

We note that Theorem 4 also holds for LLF instead of EDF. The proof can be extended in the same way as the proof for the upper bound on the speed requirement of EDF in [19].

2.6. 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 the original lower-bound instance [19], many very loose jobs, which can be scheduled on one machine, are used to delay a very tight job. To get a lower-bound instance solely consisting of tight jobs, we replace the very loose jobs with tight jobs that form a geometric structure such that they can still be scheduled on a single machine.

THEOREM 6. *For arbitrary $\alpha \in (0,1)$ and α-tight jobs,* EDF *is $\Omega(n)$-competitive, even if $m = 2$.*

Proof. We describe an instance in which release dates, deadlines, and processing times may have rational values, which can be scaled up to make parameters natural numbers again. Let $\alpha' > \alpha$ be rational. We construct n jobs, all released at time 0 such that EDF fails to feasibly schedule them on $n-1$ machines. Among the n jobs, there are $n-1$ jobs forming a geometric structure: The lengths of their intervals are $1/(1-\alpha'), \dots, 1/(1-\alpha')^{n-1}$, and their respective laxities are $1, 1/(1-\alpha'), \dots, 1/(1-\alpha')^{n-2}$. In addition to these $n-1$ jobs, there is one *critical* job of 0 laxity and the largest deadline d (i.e., $d > 1/(1-\alpha')^{n-1}$). Since the critical job has the largest deadline, EDF decides to postpone it at time 0, and it thus fails to schedule the n jobs on $n-1$ machines. Meanwhile, it is easy to verify these jobs can be feasibly scheduled on two machines since, except for the critical job, the other $n-1$ jobs can be feasibly scheduled on one machine. ∎

Using a similar idea, we can also lift up the lower bound for LLF from [19] to tight jobs. This algorithm, given m' machines, at any time t schedules m' unfinished jobs j with the smallest current laxity $\ell_j(t) = d_j - r_j - p_j(t)$, where $p_j(t)$ is the remaining processing time of job j. If only fewer jobs are available, the algorithm schedules all of them.

THEOREM 7. *Let $\alpha \in (0,1)$ be arbitrary. For α-tight jobs,* LLF *is not $f(m)$-competitive for any function f even if $m = 2$.*

In this proof, we again give w.l.o.g. rational instead of natural numbers as job parameters. The key is the following lemma, which essentially blocks one machine for a certain period.

LEMMA 8. *Let $c \in \mathbb{N}$, $\varepsilon \in (0,1)$, and $\alpha \in (0,1)$. Then there is an instance $J_{c,\varepsilon}$ consisting of α-tight jobs and a time $t^\star$ with the following properties:*

(i) *There is a feasible schedule $S^\star$ of $J_{c,\varepsilon}$ on two machines such that $p_{j'}(t^\star) = 0$ for all $j' \in J_{c,\varepsilon}$.*

(ii) *At time $t^\star$ in* LLF *given c machines, there is exactly one unfinished job $j \in J_{c,\varepsilon}$ with $d_j > t^\star$, and*

$$\frac{\ell_j(t^\star)}{d_j - t^\star} \le \varepsilon. \tag{3}$$

(iii) *Further, no job in $J_{c,\varepsilon}$ ever gets zero laxity in* LLF.

Proof. Assume w.l.o.g. that ε is rational, and let $\alpha' > \alpha$ be rational. The idea is the following: In $S^\star$, j runs uninterruptedly from 0 through $p_j =: t^\star$ on the first of the two machines (p_j and d_j will be fixed later). In LLF, however, j is delayed by *waves* of jobs, which are run on the other machine in $S^\star$. A wave released at time t of *size* β is defined as follows: There are c jobs released at time t with interval lengths $\beta \cdot (1-\alpha')^{c-1}, \beta \cdot (1-\alpha')^{c-2}, \dots, \beta$ and laxities $\beta \cdot (1-\alpha')^{c}, \dots, \beta \cdot (1-\alpha')$.

Note that, indeed, if $\ell_j(t) \geq \beta$ for all $t \leq t^\star$, a wave decreases the laxity of j by $\delta(\beta) \coloneqq \beta \cdot (1-\alpha')^{c-1} - \beta \cdot (1-\alpha')^c > 0$ in LLF. It is also easy to verify that a wave can actually be scheduled on a single machine via EDF.

We now set the parameters of j and the release times of the waves to get properties (i) and (ii). We set $\ell_j \coloneqq 1$ and wish to have $\ell_j(p_j) \leq \varepsilon$ so as to fulfill inequality (3). To do so, we release $\lceil (1-\varepsilon)/\delta(\varepsilon) \rceil$ waves of size ε. Here, we release the next wave whenever the previous wave is finished in $S^\star$ (on one machine via EDF) and in LLF. If we let p_j be the time when the last wave is finished in $S^\star$ and LLF, property (i) follows. Since inequality (3) is fulfilled and all waves are finished by LLF at time $t^\star$, Property (ii) also is fulfilled. Note that, indeed, no job in $J_{c,\varepsilon}$ ever gets zero laxity, so property (iii) is fulfilled as well. ∎

To prove the theorem, we will now apply the lemma recursively and thereby block an arbitrary number of machines.

Proof of Theorem 7. Note that the claim actually follows from a stronger version of Lemma 8: Let $c \in \mathbb{N}$, $\varepsilon \in (0,1)$, $\alpha \in (0,1)$, and $k \in \mathbb{N}$ with $k \leq n$. Then there is an instance $J^k_{c,\varepsilon}$ consisting of α-tight jobs and a *critical time* $t^\star$ with the following properties:

(i′) There is a feasible schedule $S^\star$ of $J^k_{c,\varepsilon}$ on 2 machines such that $p_j(t) = 0$ for all $j \in J^k_{c,\varepsilon}$.

(ii′) At time $t^\star$ in LLF given c machines, there are k different unfinished *critical* jobs $j \in J^k_{c,\varepsilon}$ with identical deadlines $d_j > t^\star$ and

$$\frac{\ell_j(t^\star)}{d_j - t^\star} \leq \varepsilon. \tag{4}$$

(iii′) Further, no job in $J^k_{c,\varepsilon}$ ever gets zero laxity in the LLF schedule.

We prove the claim by induction on k for arbitrary c, ε, and α. The base case is given by Lemma 8. To show that $J^k_{c,\varepsilon}$ as above exists, we first release $J_{c,\varepsilon'}$ for some ε' yet to be determined, resulting in a critical moment $t^\star$ and a critical job j. Then, at $t^\star$, we release a scaled-down version of $J^{k-1}_{c-1,\varepsilon}$ such that the deadlines of its critical jobs are exactly d_j. We set ε' such that j's laxity in LLF is always smaller than that of each job in $J^{k-1}_{c-1,\varepsilon}$, which is possible by Property (iii) of $J^{k-1}_{c-1,\varepsilon}$. This means that, in the LLF schedule, j gets a separate machine from time $t^\star$ on, and the claim follows from this fact and the induction hypothesis for $J^{k-1}_{c-1,\varepsilon}$. ∎

The remaining part of the paper is concerned with a more sophisticated algorithm for α-tight jobs.

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 9. *Let G be a set of α-tight jobs and let T be a nonempty 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 10. *If there exists a (μ,β)-critical pair, then $m \geq \frac{\mu-4-4\cdot\lceil\log 8/\beta\rceil}{8+8/\alpha\cdot\lceil\log 8/\beta\rceil} = \Omega(\frac{\mu}{\log 1/\beta})$.*

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

COROLLARY 11. *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 1.

To that end, the following simple lemma will be useful several times. A simple greedy algorithm finds such a subset. This result is surely folkloric, but as we could not find a proof, we still provide a proof.

LEMMA 12. *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'.*

Proof. Given a set of jobs S, we construct S' by a simple greedy procedure. Initially S' is an empty set. Whenever there is a time point in $I(S)$ that is not covered by a job in S', we find the smallest such time point t. Let $j \in S$ be a job covering t and having the largest deadline. Clearly, such a job exists, and we add j to S'. This procedure continues until each time point in $I(S)$ is covered by a job in S'.

By construction, each time point in $I(S)$ is covered by at least one job in S'. It remains to show that each $t \in I(S)$ is covered by at most two jobs in S'. For the sake of contradiction, suppose there are three jobs $j_1, j_2, j_3 \in S'$ (added in this order) covering t, which are added because of the uncovered time points $t_1 < t_2 < t_3$. As j_2 is added after j_1, we have $d_{j_1} < t_2$ and therefore, using that t is covered by both j_1 and j_2, we have $t < t_2$. Using this and the fact that j_3 covers t and $t_3 > t_2$, j_3 also covers t_2. Further it holds that $t_3 > d_{j_2}$, so we have $d_{j_3} > d_{j_2}$, which is a contradiction to the fact that j_2 (rather than j_3) is added by the greedy procedure to cover t_2. ∎

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

LEMMA 13. *Let (G,T) be a (μ,β)-critical pair. There exist pairwise disjoint subsets $G_1^\star, \dots, G_{\lfloor\mu/4\rfloor}^\star$ of G such that*

(i) *$T \subseteq I(G_1^\star) \subseteq \cdots \subseteq I(G_{\lfloor\mu/4\rfloor}^\star)$,*
(ii) *$\sum_{j \in G_i^\star} \ell_j \leq 4|T|/\beta$ 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 $G_i^+ \coloneqq G \setminus (G_{i+1} \cup \cdots \cup G_{\lceil\mu/2\rceil})$ using Lemma 12. 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 $T \subseteq I(G_1) \subseteq \cdots \subseteq I(G_{\lceil\mu/2\rceil})$. First, 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 12).

Second, 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 12, 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

$$|T| = \left| \bigcup_{j \in G'} (T \cap I(j)) \right| \geq \frac{\sum_{j \in G'} |T \cap I(j)|}{2\lceil \mu/2 \rceil} \geq \frac{\sum_{j \in G'} \beta \ell_j}{2\lceil \mu/2 \rceil}, \tag{5}$$

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

Now we argue that we can further choose $G_1^\star, \ldots, G_{\lfloor \mu/4 \rfloor}^\star$ from the family of job sets $G_1, \ldots, G_{\lceil \mu/2 \rceil}$ 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 jobs in 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 (5). ∎

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 14. *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 1. ∎

We are now ready to prove the main theorem.

Proof of Theorem 10. Let (G, T) be a (μ, β)-critical pair. Let $G_1^\star, \ldots, G_{\lfloor \mu/4 \rfloor}^\star$ be subsets of G that satisfy the two conditions of Lemma 13, i.e., (i) $T \subseteq I(G_1^\star) \subseteq \cdots \subseteq I(G_{\lfloor \mu/4 \rfloor}^\star)$ and (ii) $\sum_{j \in G_i^\star} \ell_j \leq 4|T|/\beta$, for all $i \in \{1, \ldots, \lfloor \mu/4 \rfloor\}$. The proof idea is to bound the contribution of certain subsets $G_q^\star, \ldots, G_{\lfloor \mu/4 \rfloor}^\star$ to the interval $I(G_q^\star)$ and show the bound on m by achieving a contradiction to the load bound in Theorem 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 $G_q^\star$ 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, $G_q^\star$ does exist, and it follows from $T \subseteq I(G_1^\star)$ and repeatedly applying Lemma 14 that

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

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

$$|I(G_q^\star)| \geq 2^k \cdot |T| \geq 2^k \cdot \frac{\beta}{4} \cdot \sum_{j \in G_i^\star} \ell_j \geq 2 \cdot \sum_{j \in G_i^\star} \ell_j, \tag{6}$$

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

For $i \geq q$, the contribution of $G_i^\star$ to $I(G_q^\star)$ can be bounded from below as follows:

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

For $i \geq q$, (6) and $I(G_q^\star) \subseteq I(G_i^\star)$ imply $C(G_i^\star, I(G_q^\star)) \geq |I(G_q^\star)|/2$.

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 $G_i^\star$ such that $C(G_i^\star, I(G_q^\star)) \geq |I(G_q^\star)|/2$. Hence,

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

This is a contradiction to Theorem 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 10 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 9 by an averaging condition.

DEFINITION 15. *Let G be a set of α-tight jobs and let T be a nonempty 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 (5) in the proof of Lemma 13.) Hence, we have the following theorem.

THEOREM 16. *If there exists a weakly (μ, β)-critical pair, then $m = \Omega(\frac{\mu}{\log 1/\beta})$.*

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 ith 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 1st, 2nd, ..., $(i - 1)$th budgets have become 0. Once $i - 1$ such jobs are already running and the currently considered job has an empty ith budget, then we process this job even if its other budgets are not empty.

We now describe our algorithm in detail. 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 1st budget of each of these jobs. As long as it is positive, we do not process the corresponding job and charge its 1st budget by the time duration until the next computation time. Once we find a job whose 1st 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 2nd

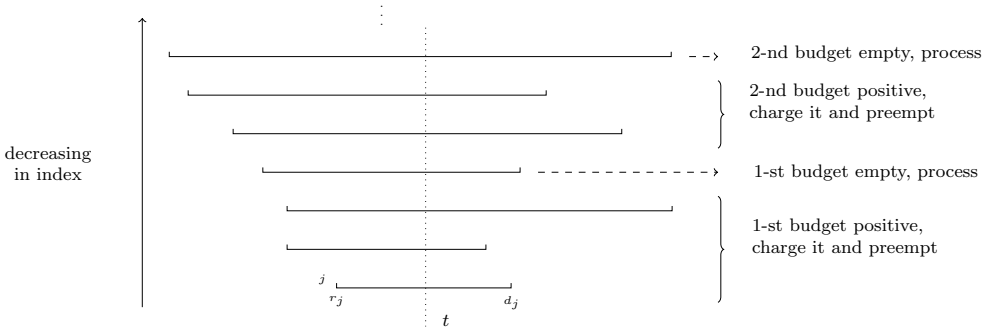

FIG. 1. *Illustration of our algorithm. At time t, we consider all relevant jobs in reverse order of their indices. After having found i active jobs, we check the $(i+1)$th budget of the current job. If this budget is not empty, we charge it and preempt/delay the job; otherwise the job becomes active.*

budget of jobs (instead of the 1st budget; because we have one active job). As long as the 2nd budgets are nonempty we delay the jobs and charge their 2nd budgets; once we find a job with an empty 2nd budget, this job becomes the second active job $a_2(t)$. Then we continue with verifying the 3rd 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 17. *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 11.

LEMMA 18. *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 $T_\mu := \{t \mid \text{the } \mu\text{th budget of } j^\star \text{ is charged at } t\}$. Moreover, for all $i = \mu - 1, \ldots, 1$, we define

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

We show that (F,T) is indeed a $(\mu, 1/\mu)$-critical pair. We first show that condition (i) in Definition 9 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 ith budget is charged at t. Notice that the ith 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 ith active job at some $t' \in T_k$ for $k > i$, where $t < t'$ because the remaining ith 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 kth 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 kth budget of j_k is charged at time t', there must exist active jobs $a_{i+1}(t'), \dots, a_{k-1}(t')$, each of which covers time t', and we have $j_k < a_h(t') < j_i$ for all $h \in \{i+1, \dots, k-1\}$. Again using $r_{a_h(t')} \le r_{j_i} \le t \le t'$, 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 ith active job at some time t. As the ith budget (which initially amounted to ℓ_j/μ) is exhausted at time t, it follows by definition of T_i that $|T_i \cap I(j)| \ge \ell_j/\mu$, and the lemma follows. ∎

We are now ready to prove the main theorem.

Proof of Theorem 17. 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 18. Now, Corollary 11 implies $\mu \le cm \log \mu$ for some constant c. Thus, there exists a constant c' such that $m' \le c' m \log m'$, i.e., the algorithm fails only if $m' \le c' m \log m'$. 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 4) and applying Theorem 3 when m is not known, 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. For example, in the proof of Theorem 10, a more careful analysis could replace $|I(G^\star_{\lceil 2m/\alpha \rceil \cdot k})| \ge 2^k |T|$ by $|I(G^\star_{\lfloor 2m/\alpha \rfloor \cdot k+1})| \ge 2^k |T|$, 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 $d_j \le d_{j'}$ for any two jobs j and j'.

THEOREM 19. *For laminar and agreeable instances, our algorithm has constant competitive ratio.*

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 μth 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 11.[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 $H_\mu \coloneqq \{j^\star\}$, $T_\mu \coloneqq \{t \mid \text{the } \mu\text{th budget of } j^\star \text{ is charged at } t\}$,

$$\begin{aligned} F_i &\coloneqq \{j \mid j = a_i(t) \text{ for some } t \in T_{i+1} \cup \cdots \cup T_\mu\}, \\ H_i &\coloneqq \{j \in F_i \mid \text{there does not exist } j' \in F_i \\ &\qquad\qquad \text{such that } I(j) \cap I(j') \neq \emptyset \text{ and } j' < j\}, \\ \text{and } T_i &\coloneqq \{t \mid \text{the } i\text{th budgetof some } j \in H_i \text{ is charged at } t\} \end{aligned}$$

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 and will be proven in subsection 6.1.

LEMMA 20. *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 20, we can prove Theorem 19 without using any further structural information.

Proof of Theorem 19. We show that our algorithm never finds a μth 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 16.

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 20 (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 20 (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 ith 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 ith 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.

[1] We are dropping constants by writing $m = \Omega(\mu / \log(1/\beta))$ in Theorem 16. The logarithm is actually taken over $8/\beta$ instead of $1/\beta$ (see (6)), and thus, $\beta = 1$ does not cause a problem in computation.

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

Case 3: We have $a_i(t) > j$. Recall that the algorithm considers jobs one by one in decreasing order of index. After it found an ith active job $a_i(t)$ at t, it will never check the ith budget of jobs of lower indices. Hence the ith 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 ith active job at some time t and thus its ith 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 restricted 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 satisfy 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 12 to get such a structure. When additionally handling loose jobs by EDF (Theorem 4), 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 3 we derive a 96-competitive algorithm for laminar instances and a 176-competitive algorithm for agreeable instances.

6.1. Proof of structural lemma. We will prove Lemma 20 separately for the laminar instances as well as agreeable instances. Before we separate the analysis into the two special cases, we state another lemma that is also independent of any structural information and will be useful in both special cases.

LEMMA 21. *Let (H, T) be a failure pair. Then we have $F_i \succ F_{i+1}$ for all $i \in \{1, \ldots, \mu - 1\}$.*

Proof. Consider an arbitrary $i \in \{1, \ldots, \mu - 1\}$. By the definition of F_i, for any $j \in F_i$ we have $j = a_i(t)$ at some time $t \in T_{i'}$ where $i' > i$. We distinguish two cases and show that in each case there exists a job in F_{i+1} with the desired properties:

Case 1: We have $i' = i + 1$. By definition of T_{i+1}, there is some $j' \in H_{i+1} \subseteq F_{i+1}$ whose $(i+1)$th budget is charged at time t, implying that $t \in I(j) \cap I(j') \neq \emptyset$. Since the algorithm goes through jobs in decreasing order of indices, it holds that $j' < j$.

Case 2: We have $i' > i + 1$. Using the definition of $T_{i'}$, there is a job $j'' \in H_{i'} \subseteq F_{i'}$ whose i'th budget is charged at t, implying the existence of an $(i + 1)$st

active job j' at time t. By definition of F_{i+1}, we have $j' \in F_{i+1}$. Notice that $t \in I(j) \cap I(j') \neq \emptyset$. As the algorithm goes through jobs in decreasing order of indices it also follows that $j' < j$.

This completes the proof. ∎

6.1.1. Laminar instances. Before we prove Lemma 20 for the laminar case, we state a simple observation and show three auxiliary lemmas for laminar instances. The following observation directly follows from the way that we index jobs and the laminarity of the instance.

Observation 22. For laminar instances and two jobs j, j', $I(j) \cap I(j') \neq \emptyset$ and $j > j'$ imply $I(j) \subseteq I(j')$.

Recall the definition of H_i. Observation 22 actually implies that the set H_i is constructed by selecting the "maximal" jobs from F_i, i.e., the jobs in F_i which are not included by any other job. The following lemma is also straightforward from the observation.

LEMMA 23. *For laminar instances, $S_1 \succ S_2$ implies that $I(S_1) \subseteq I(S_2)$.*

Proof. Consider an arbitrary job $j \in S_1$. Due to $S_1 \succ S_2$ there exists $j' \in S_2$ with $j' < j$ and $I(j) \cap I(j') \neq \emptyset$. According to Observation 22 we have $I(j) \subseteq I(j')$, hence $I(S_1) \subseteq I(S_2)$. ∎

Using Observation 22, we can also show that the $\prec$-relation on sets of jobs is transitive under laminarity.

LEMMA 24. *For laminar instances, the relation $\prec$ on sets of jobs is transitive.*

Proof. Let S_1, S_2, and S_3 such that $S_1 \succ S_2$ and $S_2 \succ S_3$. We show that also $S_1 \succ S_3$ holds: For any job $j_1 \in S_1$, there exists a job $j_2 \in S_2$ with $j_1 > j_2$ and $I(j_1) \cap I(j_2) \neq \emptyset$. Further, there is a job $j_3 \in S_3$ with $j_2 > j_3$ and $I(j_2) \cap I(j_3) \neq \emptyset$. Obviously we have $j_1 > j_3$. Further, $I(j_1) \cap I(j_3) \neq \emptyset$ since according to Observation 22, we have $I(j_1) \subseteq I(j_2) \subseteq I(j_3)$. Hence, $S_1 \succ S_3$. ∎

The third auxiliary lemma allows us to induce the relation among H_i's from the relation among F_i's, as is shown by Lemma 21.

LEMMA 25. *Consider a laminar instance and a failure pair (H, T). Then $F_i \succ F_{i'}$ implies $H_i \succ H_{i'}$.*

Proof. Consider some job $j \in H_i$. By definition of H_i, we also have $j \in F_i$. Due to $F_i \succ F_{i'}$ there is a $j' \in F_{i'}$ such that $j > j'$ and $I(j) \cap I(j') \neq \emptyset$. According to the definition of $H_{i'}$, there is a job $j'' \in H_{i'}$ with $j' \geq j''$ and $I(j') \cap I(j'') \neq \emptyset$ (indeed, job j'' could be j'). Consequently, $j > j' \geq j''$. By Observation 22, $I(j) \subseteq I(j') \subseteq I(j'')$ holds, implying that $I(j) \cap I(j'') \neq \emptyset$. Thus, $H_i \succ H_{i'}$. ∎

Now we are ready to prove Lemma 20.

Proof of Lemma 20 *for laminar instances.* We first show (i), i.e., for all $H_i, H_{i'}$ with $i < i'$ we have $H_i \succ H_{i'}$. By Lemma 24, it suffices to show $H_i \succ H_{i+1}$ for all $i = 1, \ldots, \mu - 1$. Using Lemma 25, we can even restrict to showing $F_i \succ F_{i+1}$ for all $i = 1, \ldots, \mu - 1$. This is exactly the statement of Lemma 21.

It remains to show (ii), i.e., $T \subseteq I(H_i)$ for all H_i. Towards this, we first claim that, $I(H_i) = I(F_i)$. It is easy to see that $I(H_i) \subseteq I(F_i)$ as $H_i \subseteq F_i$. Meanwhile, $I(H_i) \supseteq I(F_i)$: Consider an arbitrary job $j \in F_i$. By definition of H_i either $j \in H_i$, or there exists some $j' \in H_i$ such that $I(j) \cap I(j') \neq \emptyset$ and $j > j'$, implying that

$I(j) \subseteq I(j')$ by Observation 22. Notice that by definition $T \subseteq I(F_1)$. Hence, we have $T \subseteq I(F_1) = I(H_1) \subseteq I(H_i)$, where the last relation follows from (i) and Lemma 23.∎

6.1.2. Agreeable instances. Recall that

$$H_i = \{j \in F_i \mid \text{ there does not exist } j' \in F_i \text{ such that } I(j) \cap I(j') \neq \emptyset \text{ and } j' < j\}.$$

For agreeable instances, it is easy to see that the job in F_i with the smallest index (and hence the smallest release date) is always contained in H_i. In the following we will show that, indeed, this is the only job contained in H_i.

Proof of Lemma 20 *for agreeable instances.* We first prove (i), i.e., for all H_i and $H_{i'}$ with $i < i'$ we have $H_i \succ H_{i'}$. To do so, we make the (stronger) claim that, for every $i \in \{1, \dots, \mu\}$, the following is true: For $k \geq i$ it holds that $H_k = \{j_k\}$, where j_k is the job of smallest index in F_k. Furthermore, $H_k \succ H_{k'}$ holds for any k, k' with $i \leq k < k'$. We prove this claim by (downward) induction on i. For $i = \mu$, this claim is clear from the construction of (H, T).

For $i < \mu$, we first prove that H_i consists of the single job j_i. To this end, suppose there are two jobs $j, j' \in H_i$ where $j < j'$. By the construction of H_i, it holds that $I(j) \cap I(j') = \emptyset$ and hence $d_j \leq r_{j'}$. Since we have $F_i \succ F_{i+1}$ according to Lemma 21, there exists a job $j'' \in F_{i+1}$ with $I(j) \cap I(j'') \neq \emptyset$ and $j'' < j$. According to the induction hypothesis $j_{i+1} \leq j'' < j$, implying $d_{j_{i+1}} \leq d_j$. Hence,

$$j_\mu < \dots < j_{i+1} < j \text{ and } d_{j_\mu} \leq \dots \leq d_{j_{i+1}} \leq d_j. \tag{7}$$

It is easy to see that no time $t \geq d_j$ is covered by any job in $H_{i+1}, \dots, H_\mu$. Hence, for $t \geq d_j$ we have $t \notin T_{i+1} \cup \dots \cup T_\mu$. As $d_j \leq r_{j'}$, it follows that $j' \notin H_i$ by definition of H_i, which is a contradiction. Hence, H_i consists of only a single job. Recall that, by definition of H_i, the job j_i of smallest index in F_i is contained in H_i, and thus $H_i = \{j_i\}$.

Next we prove that $H_k \succ H_{k'}$ holds for any k, k' with $i \leq k < k'$. As we have shown (7) already, it remains to be shown that $I(j_i) \cap I(j_h) \neq \emptyset$ for any $h > i$. Suppose this is not true for some h, i.e., $d_{j_h} \leq r_{j_i}$. As j_h is an hth active job at some time t, there is also an ith active job at the same time and $a_i(t) \in F_i$ by definition of F_i. Note that, as $I(a_i(t)) \cap I(j_h) \neq \emptyset$, we have $r_{a_i(t)} < d_{j_h} \leq r_{j_i}$. Thus, $a_i(t) < j_i$ holds, contradicting the fact that j_i is the job of smallest index in F_i.

We proceed to proving (ii): For all H_i, we have $T \subseteq I(H_i)$. As (i) implies that $r_1 \geq \dots \geq r_\mu$ and $d_1 \geq \dots \geq d_\mu$, it suffices to show that $T \subseteq I(H_1) \cap I(H_\mu) = I(j_1) \cap I(j_\mu)$.

We first show that for all $t \in T$, it holds that $t \in I(j_1)$. By definition of T, the ith budget of job j_i is charged at t for some $i \in \{1, \dots, \mu\}$. If $i = 1$, clearly $t \in I(j_1)$. Otherwise, there is a 1st active job j at time t, and by definition we have $j \in F_1$. If $j = j_1$, again we are done. Otherwise, we have $j_1 < j$, and thus $r_{j_1} \leq r_j$. Now notice that $t \in I(j) \cap I(j_h)$, and thus $r_j \leq t < d_{j_h}$ holds. As $r_{j_1} \leq r_j$ and $d_{j_h} \leq d_{j_1}$ (by (i)), we have $t \in I(j_1)$.

Next, we show that for all $t \in T$, it holds that $t \in I(j_\mu)$. Again by definition of T, at time t the ith budget of j_i is charged for some $i \in \{1, \dots, \mu\}$. We prove that for $t \in T_i$, $t \in I(j_\mu)$ by (downward) induction on i. The case $i = \mu$ is obvious. For $i < \mu$ we argue in the following way: As $j_i \in F_i$, it is the ith active job at some time $t' \in T_{i+1} \cup \dots \cup T_\mu$. Since its ith budget is exhausted at time t', while at time t it is still positive, we have $t < t'$. According to the induction hypothesis, $t' \in I(j_\mu)$ holds, and consequently $t < t' \leq d_{j_\mu}$. Further, it follows by (i) that $r_{j_\mu} \leq r_{j_i} \leq t$. Thus, we have $t \in I(j_\mu)$. ∎

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 of the algorithm 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 the best possible constants.

REFERENCES

[1] Y. Azar and S. Cohen, *An improved algorithm for online machine minimization*, Oper. Res. Lett., 46 (2018), pp. 128–133.

[2] N. Bansal, T. Kimbrel, and K. Pruhs, *Speed scaling to manage energy and temperature*, J. ACM, 54 (2007), 3.

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

[4] H. Chan, T. W. Lam, and K. To, *Nonmigratory online deadline scheduling on multiprocessors*, SIAM J. Comput., 34 (2005), pp. 669–682.

[5] L. Chen, N. Megow, and K. Schewior, *The power of migration in online machine minimization*, in Proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA), Pacific Grove, CA, ACM, 2016, pp. 175–184.

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

[7] J. Chuzhoy, S. Guha, S. Khanna, and J. Naor, *Machine minimization for scheduling jobs with interval constraints*, in Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS), Rome, Italy, IEEE, 2004, pp. 81–90.

[8] M. Cieliebak, T. Erlebach, F. Hennecke, B. Weber, and P. Widmayer, *Scheduling with release times and deadlines on a minimum number of machines*, in IFIP Conference on Theoretical Computer Science (TCS), Boston, MA, Springer, 2004, pp. 217–230.

[9] N. R. Devanur, K. Makarychev, D. Panigrahi, and G. Yaroslavtsev, *Online Algorithms for Machine Minimization*, preprint, arXiv:1403.0486, 2014.

[10] L. R. Ford and D. R. Fulkerson, *Maximal flow through a network*, Canad. J. Math., 8 (1956), pp. 399–404.

[11] M. R. Garey and D. S. Johnson, *Two-processor scheduling with start-times and deadlines*, SIAM J. Comput., 6 (1977), pp. 416–426.

[12] W. A. Horn, *Some simple scheduling algorithms*, Naval Res. Logist. Q., 21 (1974), pp. 177–185.

[13] S. Im, S. Li, B. Moseley, and E. Torng, *A dynamic programming framework for non-preemptive scheduling problems on multiple machines*, in Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), San Diego, CA, Society for Industrial and Applied Mathematics, 2015, pp. 1070–1086.

[14] S. Im, B. Moseley, K. Pruhs, and C. Stein, *An $O(\log \log m)$-competitive algorithm for online machine minimization*, in IEEE Real-Time Systems Symposium (RTSS), IEEE, 2017, pp. 343–350.

[15] M.-J. Kao, J.-J. Chen, I. Rutter, and D. Wagner, *Competitive design and analysis for machine-minimizing job scheduling problem*, in International Symposium on Algorithms and Computation (ISAAC), Berlin, Heidelberg, Springer, 2012, pp. 75–84.

[16] A. J. Kleywegt, V. S. Nori, M. W. P. Savelsbergh, and C. A. Tovey, *Online resource minimization*, in Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), Baltimore, MD, Society for Industrial and Applied Mathematics, 1999, pp. 576–585.

[17] T. W. Lam and K.-K. To, *Trade-offs between speed and processor in hard-deadline scheduling*, in Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), Baltimore, MD, Society for Industrial and Applied Mathematics, 1999, pp. 623–632.

[18] C. A. Phillips, C. Stein, E. Torng, and J. Wein, *Optimal time-critical scheduling via resource augmentation*, in Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing (STOC), El Paso, TX, ACM, 1997, pp. 140–149.

[19] C. A. Phillips, C. Stein, E. Torng, and J. Wein, *Optimal time-critical scheduling via resource augmentation*, Algorithmica, 32 (2002), pp. 163–200.

[20] K. Pruhs, *Collection of open problems in scheduling*, in Dagstuhl Scheduling Seminar, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Germany, 2010.

[21] B. Saha, *Renting a cloud*, in IARCS Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS), Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2013, pp. 437–448.

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

[23] G. Yu and G. Zhang, *Scheduling with a minimum number of machines*, Oper. Res. Lett., 37 (2009), pp. 97–101.