Maximizing Online Utilization with Commitment

Chris Schwiegelshohn · Uwe Schwiegelshohn

Received: date / Accepted: date

Abstract We investigate online scheduling with commitment for parallel identical machines. Our objective is to maximize the total processing time of accepted jobs. As soon as a job has been submitted, the commitment constraint forces us to decide immediately whether we accept or reject the job. Upon acceptance of a job, we must complete it before its deadline $d$ that satisfies $d \geq (1 + \varepsilon) \cdot p + r$, with $p$ and $r$ being the processing time and the submission time of the job, respectively while $\varepsilon > 0$ is the slack of the system. Since the hard case typically arises for near-tight deadlines, that is $\varepsilon \to 0$, we consider $\varepsilon \leq 1$. We use competitive analysis to evaluate our algorithms. While there are simple online algorithms with optimal competitive ratios for the single machine model, little is known for parallel identical machines.

Our first main contribution is a deterministic preemptive online algorithm with an almost tight competitive ratio on any number of machines. For a single machine, the competitive factor matches the optimal bound $\frac{1 + \varepsilon}{\varepsilon}$ of the greedy acceptance policy. Then the competitive ratio improves with an increasing number of machines and approaches $(1 + \varepsilon) \cdot \ln \frac{1}{\varepsilon}$ as the number of machines converges to infinity. This is an exponential improvement over the greedy acceptance policy for small $\varepsilon$.

In the non-preemptive case, we present a deterministic algorithm on $m$ machines with a competitive ratio of $1 + m \cdot \left( \frac{1 + \varepsilon}{\varepsilon} \right)^{\frac{1}{m}}$. This matches the optimal bound of $2 + \frac{1}{\varepsilon}$ of the greedy acceptance policy for a single machine while it again guarantees an exponential improvement over the greedy acceptance policy for small $\varepsilon$ and large $m$. In addition, we determine an almost tight lower bound that approaches $m \cdot \left( \frac{1}{\varepsilon} \right)^{\frac{1}{m}}$ for large $m$ and small $\varepsilon$.

Keywords Online scheduling · commitment · competitive analysis

Mathematics Subject Classification (2010) 68W27 · 68W40

1 Introduction

We consider an online resource allocation and scheduling problem modeled as follows. Given a set of $m$ identical machines, we process a sequence of jobs in an online fashion. Each job has a release date $r_j$, a deadline $d_j$, and a processing time $p_j$. Our objective is to maximize the total utilization of the system, that is the sum of the processing times of all completed jobs, subject to a commitment guarantee. Due to this guarantee we must decide immediately upon submission of a job, whether to accept or reject it. In case of acceptance, we must complete it before its deadline and cannot accept any other job that would prevent us from doing so.

This problem has received a significant amount of attention and tight algorithms with competitive ratios depending on $\Delta = \log \frac{\max_j p_j}{\min_j p_j}$ are known for a single machine without further assumptions on the input [2441]. To obtain more meaningful results, a number of papers use a slack parameter $\varepsilon > 0$ and require $d_j \geq (1 + \varepsilon) \cdot p_j + r_j$ for all jobs. Typically, the hard case arises for small slack ($\varepsilon \to 0$).

We consider the preemptive and the non-preemptive versions of the problem for small slack values $\varepsilon \leq 1$. In case of preemption, we allow migration and assume no penalty for preemption and migration.
We use the common approach of competitive analysis to measure the performance of online algorithms, that is, we determine the competitive ratio. Since we are facing a maximization problem, the strict competitive ratio is the maximum value of \( \frac{\text{OPT}(f)}{\text{ALG}(f)} \) over all input sequences \( f \), with \( \text{OPT}(f) \) and \( \text{ALG}(f) \) being the objective values of an optimal offline algorithm and the examined online algorithm, respectively.

1.1 Our Contribution

We show upper and lower bounds on the competitive ratio of utilization maximization with commitment in dependence of the slackness parameter \( \varepsilon \). Specifically:

- For \( m \) machines supporting preemption and migration, we give a deterministic algorithm with a strict competitive ratio of
  \[
  m \cdot (1 + \varepsilon) \cdot \left( \frac{1 + \varepsilon}{\varepsilon} \right)^{-\frac{1}{m}} - 1.
  \]
  This bound decreases with increasing \( m \), approaching \((1 + \varepsilon) \cdot \ln(\frac{1 + \varepsilon}{\varepsilon})\) as \( m \) tends to infinity and matching the competitive ratio \( \frac{1}{\varepsilon} \) of the optimal greedy algorithm for \( m = 1 \).
- We provide an almost matching lower bound of
  \[
  [m \cdot (1 + \varepsilon)] \cdot \left( \frac{1 + \varepsilon}{\varepsilon} \right)^{-\frac{1}{m}} - 1.
  \]
- For \( m \) machines without preemption, we give a deterministic algorithm with a strict competitive ratio of
  \[
  m \cdot \left( \frac{1 + \varepsilon}{\varepsilon} \right)^{\frac{1}{m}} + 1.
  \]
  For \( m > \ln \frac{1 + \varepsilon}{\varepsilon} \), we apply partitioning to improve this competitive ratio. In particular, we obtain a competitive ratio \( \varepsilon \ln(\frac{1 + \varepsilon}{\varepsilon}) + 1 \) for integral values \( \ln(\frac{1 + \varepsilon}{\varepsilon}) \) and \( m = i \cdot \ln(\frac{1 + \varepsilon}{\varepsilon}) \) for all small \( \varepsilon \), these results significantly improve over the state of the art greedy algorithm with a competitive ratio of \( 2 + \frac{1}{\varepsilon} \).
- For the non-preemptive machine environment, we also give the first lower bounds that are asymptotically greater than 1 for any number of machines. In particular, for large \( m \) and \( \varepsilon \leq 1 \), our lower bound approaches
  \[
  m \cdot \left( \frac{1}{\varepsilon} \right)^{\frac{1}{m}}
  \]
  which is almost tight (up to a \( (1 + \varepsilon)^{1/m} \) factor and the additive value 1).

1.2 Related Work

Using the three field notation of Graham et al. [1], the \( P|\text{pmtn},r_j|\sum(1 - U_j)w_j \) problem admits a pseudopolynomial time algorithm, see Lawler [20], and a FPTAS due to a technique by Pruhs and Woeginger [25]. We are not aware of any simplifications when assuming \( w_j = p_j \). For preemption without migration, these ideas were further extended to admit a \( 2 + \varepsilon \) approximation [2]. A reduction from preemption to non-preemptive due to Kalyanasundaram and Pruhs [16] yields a \( 6 + \varepsilon \) approximation. Recently, Alon et al. [1] analyzed the gap between schedules with a bounded number of \( k \) preemptions and an unbounded number of preemptions. They show that the utilization ratio is at most \( O(\log_{k+1} n, \log_{k+1} \frac{\max p_j}{\min w_j}) \), which is asymptotically optimal.

Variants of the online problem have been studied in real-time scheduling. Without commitment and assuming preemptions, there exists a tight 4-competitive deterministic algorithm on a single machine for a class of well-behaved pay-off functions including \( w_j = p_j \), see, for instance, Baruah et al. [5] and Wöginger [27]. Kalyanasundaram and Pruhs [17] gave a \( O(1) \) competitive randomized algorithm for maximizing the number of completed jobs and showed that no constant competitive deterministic algorithm exists. For general pay-off functions in a non-preemptive environment, several authors [19,24] showed finite competitive ratios of \( O(\min(\log_{\max p_j} \log_{\max w_j}, \log_{\min w_j} d_j)) \) if \( d_j = r_j + p_j \) holds. For slack \( \varepsilon > 0 \), Lucier et al. [23] proposed an algorithm with competitive ratio \( 2 + \Theta \left( \frac{1}{\sqrt{1+\varepsilon^2}} \right) + \Theta \left( \frac{1}{\sqrt{1+\varepsilon^2}-1} \right) \).

Commitments make the problem much more difficult. Indeed, Lucier et al. [23] showed that even with slack, no performance guarantees are possible for general weights \( w_j \). By proposing a model with limited commitment called \( \beta \)-responsive, Azar et al. [3] gave an algorithm with competitive ratio \( 5 + \Theta \left( \frac{1}{\sqrt{1+\varepsilon^2}} \right) + \Theta \left( \frac{1}{\sqrt{1+\varepsilon^2}-1} \right) \) for \( \varepsilon > 3 \). Similar relaxed commitment models were recently considered by Chen et al. [5] for the problem of maximizing the number of accepted jobs (\( w_j = 1 \)). For \( w_j = p_j \) (the utilization maximization problem), a number of algorithms with commitment have been proposed. On a single machine without assuming slack or preemption, Lipton and Tomkins [22] gave a lower bound of \( \Omega(\log \Delta) \) for the competitive factor of any randomized online algorithm. Additionally assuming \( d_j = p_j + r_j \) for all jobs, they obtained an upper bound of \( O(\log^{1+\delta} \Delta) \) for any \( \delta > 0 \). For arbitrary deadlines, Goldman, Parwatikar, and Suri [11] presented a \( 6(\log \Delta + 1) \) competitive algorithm.

Baruah and Haritsa [4] were among the first to present results for problems with slack when they addressed the single machine problem with preemption and commitment. For

\[\text{In literature, the stretch } f = 1 + \varepsilon \text{ is often used instead. The two notions are equivalent.}\]
this problem, they gave an algorithm with a competitive ratio \(1 + \frac{\varepsilon}{m}\) and a lower bound of \(1 + \frac{1}{m}\). For the corresponding non-preemptive problem on a single machine, Goldwasser’s algorithm [14] guarantees a tight deterministic competitive ratio of \(2 + \frac{1}{m}\). A similar competitive ratio is also given by Garay et al. [10].

For parallel identical machines, DasGupta and Palis [9] consider preemption without migration and suggest a simple greedy algorithm. They claim the same competitive factor as for the single machine problem with preemption. They also give a lower bound of \(1 + \frac{1}{me}\). For the non-preemptive case, Kim and Chwa [18] apply Goldwasser’s algorithm [12, 13] to parallel identical machines. Lee [21] considers a model without commitment and presents a non-preemptive algorithm. They give a lower bound of \(1 + \frac{\varepsilon}{m}\) for the single machine problem with preemption. They also give a lower bound of \(1 + \frac{1}{m}\). For the non-preemptive case, we consider preemption if there is a slack condition \(d_j - r_j \geq 1 + \varepsilon \cdot p_j\). We are interested in problems with a small slack (\(\varepsilon \leq 1\)). The job sequence \(J\) defines an order of the jobs such that job \(J_i\) precedes job \(J_j\) for \(r_i < r_j\).

We discuss an online scenario with commitment: the jobs arrive in sequence order and we must decide whether to accept the most recent job or not before seeing any job at a later position in the sequence. Complying with the common notation in scheduling theory, we use the binary variable \(U_j\) to express this decision for job \(J_j\): \(U_j = 0\) denotes acceptance of job \(J_j\) while the rejection of \(J_j\) produces \(U_j = 1\). We commit ourselves to complete every accepted job not later than its deadline and say that a schedule is valid if it satisfies this commitment. Therefore, we can only accept a new job if there is a valid extension of the current schedule that includes this job. We consider preemptive and non-preemptive variants of this problem. If the system allows preemption (pvmnt) then we can interrupt the execution of any job and immediately or later resume it on a possibly different machine without any penalty. In the non-preemptive case, we combine the acceptance commitment with a machine and start time commitment (immediate decision, see Bund and Goldwasser [6]).

For a previously released job, we only consider its remaining processing time when generating or modifying a preemptive schedule. Therefore, progression of time may change the values of variables that depend on the processing time. For such a variable, we use the appendix \(\lfloor t \rfloor\) to indicate its value at time \(t\).

It is our goal to maximize the total processing time of all accepted jobs (\(\sum p_j \cdot (1 - U_j)\)). An online algorithm \(ALG\) determines \(U_j(\text{ALG})\) for all jobs in \(J\) such that there is a valid schedule \(S\) for all jobs \(J_j\) with \(U_j(\text{ALG}) = 0\). \(P[t, t'](S, J)\) denotes the total processing time of schedule \(S\) for job sequence \(J\) in interval \([t, t']\) while \(P'[t, t'](J)\) is the maximum total processing time of any optimal schedule for the same job sequence in the same interval. We evaluate our algorithm by determining bounds for the competitive ratio, that is, the largest ratio between the optimal objective value and the objective value produced by online algorithm \(ALG\) for all possible job sequences \(J\): \(ALG\) has a strict competitive ratio \(c_{ALG}\) if \(c_{ALG} \geq \frac{\max\{P'[t, t'](J)\}}{\max\{P'[t, t'](J)\}}\) holds for any job sequence \(J\). Finally, \(c\) is a lower bound of the competitive ratio if \(c \leq c_{ALG}\) holds for any online algorithm \(ALG\).

2 Notations and Basic Properties

We schedule jobs of a job sequence \(J\) on \(m\) parallel identical machines (\(P_m\) model). Release date \(r_j\), processing time \(p_j\), and deadline \(d_j\) of a job \(J_j \in J\) obey the slack condition \(d_j - r_j \geq 1 + \varepsilon \cdot p_j\). \(\varepsilon > 0\) is the fixed slack factor of the system. We say that job \(J_j\) has a slack if \(d_j - r_j = (1 + \varepsilon) \cdot p_j\) holds. In the three field notation, the constraint \(\varepsilon\) indicates a problem with slack factor \(\varepsilon\). We are interested in problems with a small slack (\(\varepsilon \leq 1\)). The job sequence \(J\) defines an order of the jobs such that job \(J_i\) precedes job \(J_j\) for \(r_i < r_j\).

We discuss an online scenario with commitment: the jobs arrive in sequence order and we must decide whether to accept the most recent job or not before seeing any job at a later position in the sequence. Complying with the common notation in scheduling theory, we use the binary variable \(U_j\) to express this decision for job \(J_j\): \(U_j = 0\) denotes acceptance of job \(J_j\) while the rejection of \(J_j\) produces \(U_j = 1\). We commit ourselves to complete every accepted job not later than its deadline and say that a schedule is valid if it satisfies this commitment. Therefore, we can only accept a new job if there is a valid extension of the current schedule that includes this job. We consider preemptive and non-preemptive variants of this problem. If the system allows preemption (pvmnt) then we can interrupt the execution of any job and immediately or later resume it on a possibly different machine without any penalty. In the non-preemptive case, we combine the acceptance commitment with a machine and start time commitment (immediate decision, see Bund and Goldwasser [6]).

For a previously released job, we only consider its remaining processing time when generating or modifying a preemptive schedule. Therefore, progression of time may change the values of variables that depend on the processing time. For such a variable, we use the appendix \(\lfloor t \rfloor\) to indicate its value at time \(t\).

It is our goal to maximize the total processing time of all accepted jobs (\(\sum p_j \cdot (1 - U_j)\)). An online algorithm \(ALG\) determines \(U_j(\text{ALG})\) for all jobs in \(J\) such that there is a valid schedule \(S\) for all jobs \(J_j\) with \(U_j(\text{ALG}) = 0\). \(P[t, t'](S, J)\) denotes the total processing time of schedule \(S\) for job sequence \(J\) in interval \([t, t']\) while \(P'[t, t'](J)\) is the maximum total processing time of any optimal schedule for the same job sequence in the same interval. We evaluate our algorithm by determining bounds for the competitive ratio, that is, the largest ratio between the optimal objective value and the objective value produced by online algorithm \(ALG\) for all possible job sequences \(J\): \(ALG\) has a strict competitive ratio \(c_{ALG}\) if \(c_{ALG} \geq \frac{\max\{P'[t, t'](J)\}}{\max\{P'[t, t'](J)\}}\) holds for any job sequence \(J\). Finally, \(c\) is a lower bound of the competitive ratio if \(c \leq c_{ALG}\) holds for any online algorithm \(ALG\).

3 Online Scheduling with Preemption

We present an approach for solving our problem on parallel identical machines with preemption. Contrary to previous approaches, our algorithm is based on lazy acceptance, that is, we may not accept a job although its acceptance allows a valid schedule. We show that our lazy acceptance leads to better competitive ratios than the greedy acceptance approach.

In this section, we frequently use function \((1 + \varepsilon)^x\) with the identity

\[
\sum_{j=i+1}^{m+i} \left(\frac{1 + \varepsilon}{\varepsilon}\right)^{\frac{j}{\varepsilon}} = \sum_{j=i}^{m+i-1} \left(\frac{1 + \varepsilon}{\varepsilon}\right)^{\frac{j}{\varepsilon}} + \frac{1}{\varepsilon}\left(\frac{1 + \varepsilon}{\varepsilon}\right)^{\frac{i}{\varepsilon}}
\]

(1)

and the threshold expression

\[
f(m, \varepsilon) := \frac{\varepsilon}{1 + \varepsilon} \sum_{j=0}^{m-1} \left(\frac{1 + \varepsilon}{\varepsilon}\right)^{\frac{j}{\varepsilon}} = \frac{1}{1 + \varepsilon} \cdot \frac{1}{\left(\frac{1 + \varepsilon}{\varepsilon}\right)^{\frac{m}{\varepsilon}} - 1}.
\]

(2)

We start by recalling a known aggregation function for a sequence \(J\) of jobs that are submitted at time \(t\) or earlier but not yet completed (\(p_j \cdot (1 - U_j)\)): \(V_{\min}(\tau)\) for \(\tau \geq t\) is the minimum total processing time that we must execute in interval \([t, \tau]\) of any valid schedule.

\[
V_{\min}(\tau)|t := \sum_{j: J \not\subseteq J \text{l}} \left\{ \begin{array}{ll}
0 & \text{for } d_j - p_j \geq \tau \\
\tau - (d_j - p_j) & \text{for } \tau \geq d_j \\
\tau & \text{else}
\end{array} \right.
\]

Based on function \(V_{\min}\), Horn [15] has given a necessary and sufficient condition for the existence of a valid preemptive schedule:
Theorem 1 (Horn) There is a valid preemptive schedule in interval \( [t, \max_j \{ f_j \} ] \) for a sequence \( J \) of jobs with deadlines on \( m \) parallel identical machines if and only if we have
\[
d_j \geq t + p_j \quad \text{for every job } J_j \quad \text{with } p_j \geq 0 \quad \text{and} \quad V_{\min}(\tau)_{\mid J \in J} \leq (\tau - t) \cdot m \quad \text{for every } \tau \in (t, \max d_j]. \tag{3}
\]

Proof See Horn [15], proof of Theorem 2.

Next we introduce two algorithms: the first algorithm decides whether to accept a newly submitted job while the second algorithm determines the schedule of the accepted jobs. Afterwards, we prove that the combination of both algorithms always produces a valid solution and give the competitive ratio of our approach.

Main component of Algorithm 1. Acceptance for Preemption is the non decreasing threshold deadline \( d_{\min} \). We simply accept a new job if and only if its deadline is at least as large as \( d_{\min} \), see Lines 5 to 6 and 8 to 9. Initially, we assume \( r_j \geq 0 \) for all jobs and set \( d_{\min} = 0 \) (Line 1). Whenever there is a submission of a new job then Line 3 prevents \( d_{\min} \) to be in the past. Line 4 introduces a compensation value \( V_A \) for job parts that is scheduled earlier due to the availability of machines, compare Algorithm 2 lines 14-18, and Lemma 1. Due to this early scheduling, progress of time may result in \( V_{\min}(d_{\min})_{\mid r_j} < (d_{\min} - r_j) \cdot f(m, \varepsilon) \). \( V_A \) compensates for this reduction and prevents an increase of the threshold that is too small. To this end, Line 7 sets \( d_{\min} \) to the largest value \( \tau \) that satisfies \( V_{\min}(\tau)_{\mid r_j} + V_A = (\tau - r_j) \cdot f(m, \varepsilon) \).

Algorithm 1: Acceptance for Preemption

\[
\begin{align*}
1: & \quad d_{\min} = 0 \\
2: & \quad \text{for each newly submitted job } J_j \text{ do} \\
3: & \quad d_{\min} = \max(d_{\min}, r_j) \\
4: & \quad V_A = (d_{\min} - r_j) \cdot f(m, \varepsilon) - V_{\min}(d_{\min})_{\mid r_j} \\
5: & \quad \text{if } d_j \geq d_{\min} \text{ then} \\
6: & \quad \text{accept } J_j \\
7: & \quad d_{\min} = \arg \max \{(\tau - r_j) \cdot f(m, \varepsilon) = V_{\min}(\tau)_{\mid r_j} + V_A \} \\
8: & \quad \text{else} \\
9: & \quad \text{reject } J_j \\
10: & \quad \text{end if} \\
11: & \quad \text{end for}
\end{align*}
\]

Algorithm 2 uses the current time \( t \) as input parameter and generates a schedule for interval \( [t, T] \) including the specification of the end of the interval \( T \) (Lines 6, 11, and 17). We must run the algorithm again if we have accepted a new job or as soon as time has progressed to \( T \). We determine the largest deadline \( d_k \) such that at most \( m \) jobs contribute to \( V_{\min}(d_k)_{\mid r} \) (Line 1). If there is no such deadline, we skip any preallocation and select the setting to produce an LRPT schedule for the contributions of the jobs to \( V_{\min}(d_k)_{\mid r} \) (Lines 3 to 7). Otherwise, we execute a preallocation by starting each of the jobs contributing to \( V_{\min}(d_k)_{\mid r} \) on a separate machine (Lines 9 and 10). \( T \) exceeds \( t \) by at most the smallest contribution among those jobs to \( V_{\min}(d_k)_{\mid r} \) (Line 11). \( m_{LRPT} \) is the number of idle machines after this preallocation (Line 12). If there are idle machines in interval \( [t, T] \) and the total number of uncompleted jobs exceeds \( m \) then we apply LRPT to determine a schedule starting at time \( t \) on these \( m_{LRPT} \) idle machines for the contribution of any not yet allocated job to \( V_{\min}(d_{k+1})_{\mid r} \) (Lines 15 and 16). If a machine becomes idle in this LRPT schedule before the previously determined value of \( T \) then we reduce \( T \) accordingly (Line 17). Since Algorithm 2 prioritizes contributions to \( V_{\min}(t)_{\mid r} \) over additional contributions to \( V_{\min}(r)_{\mid r} \), for \( t_2 > t_1 > t \), it generates a valid schedule if one exists, see Theorem 1.

Next we discuss the influence of progression of time due to Algorithm 2 on \( V_{\min} \) and on \( d_{\min} \) for a valid schedule.

Lemma 1 If Algorithm 2 has generated a valid schedule starting at time \( t \) then progression from time \( t \) to time \( t' \) without acceptance of a new job in interval \( (t, t') \) produces
\[
V_{\min}(\tau)_{\mid r} \leq \frac{\tau - t'}{\tau - t} \cdot V_{\min}(\tau)_{\mid r} \quad \text{for each } \tau \geq t'
\]
and for each \( \tau > d_{\min} \),
\[
V_{\min}(\tau)_{\mid r} - V_{\min}(d_{\min})_{\mid r} < (\tau - d_{\min}) \cdot f(m, \varepsilon).
\]

Proof Eq. (4) holds for time \( \tau > t' \) if no machine is idle in interval \( [t, t'] \) and every machine only executes job parts contributing to \( V_{\min}(\tau)_{\mid r} \) due to
\[
\frac{V_{\min}(\tau)_{\mid r}}{\tau - t'} \geq \frac{V_{\min}(\tau)_{\mid r} - m \cdot (t' - t)}{\tau - (t' - t)} = \frac{V_{\min}(\tau)_{\mid r}}{\tau - t'}.
\]
If at least one machine executes in interval \([t, t')\) some job parts that do not contribute to \(V_{\text{min}}(\tau)|_t\) for some time \(\tau \geq t'\) or has some idle time then Algorithm 2 guarantees that at time \(t'\), there are \(k < m\) jobs with uncompleted job parts contributing to \(V_{\text{min}}(\tau)|_{t'}\) and the total amount of processing time in interval \([t, t')\) that contributes to \(V_{\text{min}}(\tau)|_{t'}\) is at least \(k \cdot (t' - t)\). Then we have

\[
\frac{V_{\text{min}}(\tau)|_t}{\tau - t} \geq \frac{V_{\text{min}}(\tau)|_{t'} + k \cdot (t' - t)}{\tau - t' + t' - t} \geq \frac{V_{\text{min}}(\tau)|_{t'}}{\tau - t'}.
\]

We assume that Eq. (5) holds for \(t\) and apply an inductive approach. For \(d_{\text{min}}|_t \geq t'\), Algorithm 1 sets \(d_{\text{min}}|_{t'} = d_{\text{min}}|_t\). Due to Algorithm 2 we have for each \(\tau > d_{\text{min}}|_{t'}\),

\[
V_{\text{min}}(\tau)|_{t'} - V_{\text{min}}(d_{\text{min}}|_{t'}) \leq V_{\text{min}}(\tau)|_{t'} - V_{\text{min}}(d_{\text{min}}|_{t'}),
\]

\[
< (\tau - d_{\text{min}}|_{t'}) \cdot f(m, \varepsilon),
\]

\[
< (\tau - d_{\text{min}}|_{t'}) \cdot f(m, \varepsilon).
\]

For \(d_{\text{min}}|_{t'} < t'\), Algorithm 1 sets \(d_{\text{min}}|_{t'} = t'\) and \(V_{\Delta}|_{t'} = 0\). Then we have for each \(\tau > d_{\text{min}}|_{t'}\),

\[
V_{\Delta}|_{t'} + V_{\text{min}}(\tau)|_{t'} = V_{\text{min}}(\tau)|_{t'} = \frac{\tau - t'}{\tau - t} \cdot V_{\text{min}}(\tau)|_{t'}
\]

\[
\leq \frac{\tau - t'}{\tau - t} \cdot (V_{\text{min}}(\tau)|_{t'} + V_{\Delta}|_{t'})
\]

\[
< \frac{\tau - t'}{\tau - t} \cdot ((\tau - t) \cdot f(m, \varepsilon))
\]

\[
< (\tau - d_{\text{min}}|_{t'}) \cdot f(m, \varepsilon).
\]

Next, we prove that Algorithms 1 and 2 guarantee the existence of a valid schedule. To this end, we introduce a new function \(V(x)\) with \(0 \leq x\). In the middle part of this definition, we use \(x = y \cdot \left(\frac{m}{\min(t, t')}ight)\) with \(h \in \{0, 1, \ldots, m - 1\}\) and \(1 \leq y \leq \left(\frac{1 + \varepsilon}{\min(t, t')}\right)^h\).

\[
V(x) := \begin{cases} \frac{x \cdot f(m, \varepsilon)}{1 + \varepsilon} & \text{for } x \geq 1 \\ \sum_{i=0}^{h} \left(\frac{x}{1 + \varepsilon}\right)^{\min(t, t')} \cdot x \cdot (m - h - 1) & \text{for } \frac{x}{1 + \varepsilon} \geq x \geq 0 \end{cases}
\]

Eq. (9) holds since the slopes are non-negative and decreasing in interval \((0, 1)\). Based on function \(V(x)\), we claim the following property of our combination of algorithms:

**Lemma 2 Algorithms**

**Algorithm 1 and 2** guarantee

\[
V_{\text{min}}(\tau)|_{t'} \leq (d_{\text{min}} - t) \cdot f\left(\frac{\tau - t}{d_{\text{min}} - t}\right) \quad \text{for } d_{\text{min}} > \tau \geq t
\]

and for \(\tau \geq d_{\text{min}}\),

\[
V_{\text{min}}(\tau)|_{t'} \leq V_{\text{min}}(d_{\text{min}})|_{t'} + (\tau - d_{\text{min}}) \cdot f(m, \varepsilon).
\]

Proof We use an inductive approach by assuming validity of Eq. (10) and (11) for \(t\) and separately consider the events progression of time and acceptance of a new job.

We start with progression from time \(t\) to time \(t'\) without acceptance of a new job in interval \((t, t')\). Eq. (10) and (11) clearly hold for time \(t = t'\). Eq. (10) guarantees validity of Eq. (11) for \(\tau > d_{\text{min}}|_{t'} = \max\{d_{\text{min}}|_{t'}, t'\}\). For \(t' < \tau \leq d_{\text{min}}|_{t'} = d_{\text{min}}|_{t'}\), Eq. (10) also remains valid due to Lemma 1.

\[
V_{\text{min}}(\tau)|_{t'} \leq \frac{\tau - t'}{\tau - t} \cdot V_{\text{min}}(\tau)|_{t'}
\]

\[
\leq \left(\frac{\tau - t'}{\tau - t}\right) \cdot d_{\text{min}}|_{t'} - t \cdot f\left(\frac{\tau - t}{d_{\text{min}}|_{t'} - t}\right)
\]

\[
\leq (d_{\text{min}}|_{t'} - t) \cdot f\left(\frac{\tau - t}{d_{\text{min}}|_{t'} - t}\right)
\]

The second event is the acceptance of a new job \(J_j\) with \(r_j = t, (1 + \varepsilon) \cdot p_j \leq d_j - r_j,\) and \(d_j \geq d_{\text{min}}\), see Algorithm 1.

To simplify the formal description of this event, we omit the appendix \(|_t\) of the variables and apply a time shift by \(-t\), that is, we assume \(r_j = t = 0\). This time shift does not affect the validity of our transformations since they always use \(d_{\text{min}}|_{t'} - t\) or \(\tau - t\).

Again we assume the validity of Eq. (10) and (11) before the submission of \(J_j\). The acceptance of \(J_j\) produces a new deadline threshold \(d_{\text{new}}|_{d_{\text{min}}|_{t'}} \geq d_{\text{min}}|_{t'}\). We partition time into several intervals: Eq. (11) is always valid for \(\tau \geq d_{\text{new}}|_{d_{\text{min}}|_{t'}}\) due to Line 7 of Algorithm 1 and it remains valid for \(d_{\text{new}}|_{d_{\text{min}}|_{t'}}\),

\[
\left(\frac{\tau - t'}{\tau - t}\right) \cdot V_{\text{min}}(\tau)|_{t'} \leq V_{\text{min}}(\tau)|_{t'}
\]

\[
\leq (d_{\text{min}}|_{t'} - t'') \cdot f\left(\frac{\tau - t'}{d_{\text{min}}|_{t'} - t''}\right)
\]

For the remaining intervals, we start with the case \(d_j = d_{\text{min}} = V_{\text{min}}(d_{\text{min}}) + V_{\Delta}\).
If job \( J_j \) has tight slack (\( p_j = \frac{d_{min}}{1+\epsilon} \)) then Line 7 of Algorithm 1 produces

\[
d_{min}^{new} \geq \frac{V_{new}(d_{min}) + V_A}{f(m, \epsilon)} = \frac{V_{min}(d_{min}) + V_A + p_j}{f(m, \epsilon)} = d_{min} + \frac{d_{min}}{(1+\epsilon) \cdot f(m, \epsilon)} \tag{12}
\]

For \( \tau = d_{min} \cdot y \cdot \left( \frac{m-h}{m} \right) \) with \( h \in \{0,1, \ldots, m-1\} \) and \( 1 \leq y \leq (\frac{1+\epsilon}{\epsilon})^\frac{1}{h} \), there is

\[
V_{new}^{min}(\tau) = V_{min}(\tau) + \tau - d_{min} \cdot \frac{\epsilon}{1+\epsilon} \leq d_{min} \cdot \sum_{i=0}^{h} \left( \frac{\epsilon}{1+\epsilon} \right)^{\frac{m-h}{m}} + d_{min} \cdot y \cdot (m-h-1) \cdot \left( \frac{\epsilon}{1+\epsilon} \right)^{\frac{m-h}{m}} + d_{min} \cdot y \cdot (m-h) \cdot \left( \frac{\epsilon}{1+\epsilon} \right)^{\frac{m-h}{m}}. \tag{14}
\]

For \( h = 0 \), that is \( d_{min} \cdot \frac{\epsilon}{1+\epsilon} \leq \tau \leq d_{min} \cdot \left( \frac{1+\epsilon}{\epsilon} \right)^\frac{1}{h} \leq d_{min} \cdot \frac{\epsilon}{1+\epsilon} \), we have

\[
V_{new}^{min}(\tau) \leq \tau \cdot m = \tau \cdot d_{min} \cdot \left( \frac{1+\epsilon}{\epsilon} \right)^\frac{1}{h} \leq d_{min} \cdot \left( \frac{1+\epsilon}{\epsilon} \right)^\frac{1}{h} \cdot V \left( \frac{\tau}{d_{min} \cdot \left( \frac{1+\epsilon}{\epsilon} \right)^\frac{1}{h}} \right) \leq d_{min} \cdot \left( \frac{1+\epsilon}{\epsilon} \right)^\frac{1}{h} \cdot V \left( \frac{\tau}{d_{min} \cdot \left( \frac{1+\epsilon}{\epsilon} \right)^\frac{1}{h}} \right) \tag{13}
\]

For \( h \in \{1, \ldots, m-1\} \), there is

\[
V_{new}^{min}(\tau) = V_{min}(\tau) + \tau - d_{min} \cdot \frac{\epsilon}{1+\epsilon} \leq d_{min} \cdot \left( \frac{1+\epsilon}{\epsilon} \right)^\frac{1}{h} \cdot V \left( \frac{\tau}{d_{min} \cdot \left( \frac{1+\epsilon}{\epsilon} \right)^\frac{1}{h}} \right) + d_{min} \cdot y \cdot (m-h) \cdot \left( \frac{\epsilon}{1+\epsilon} \right)^{\frac{m-h}{m}} \tag{15}
\]

We compare this tight slack scenario to the acceptance of a job with \( d_j = d_{min} \) but without tight slack (\( p_j < \frac{d_{min}}{1+\epsilon} \)). We indicate the variables of the non-tight scenario with \( \tilde{\tau} \) and obtain for \( d_j - p_j < \tau \leq d_{min} \)

\[
V_{new}^{min}(\tau) = V_{new}(\tau) \leq d_{min} \cdot \left( \frac{1+\epsilon}{\epsilon} \right)^\frac{1}{h} \cdot V \left( \frac{\tau}{d_{min} \cdot \left( \frac{1+\epsilon}{\epsilon} \right)^\frac{1}{h}} \right) \tag{16}
\]

This result leads to

\[
(V_{new}^{min}(\tau))' \leq \left( V_{min}(d_{min}) \right)' + V_A \cdot V_{new}(\tau) \leq \left( V_{min}(d_{min}) \right)' + V_A \cdot V_{new}(\tau) \leq \left( V_{min}(d_{min}) \right)' + V_A \cdot \left( \frac{\tau}{d_{min} \cdot \left( \frac{1+\epsilon}{\epsilon} \right)^\frac{1}{h}} \right) \tag{17}
\]

For the general case \( p_j \leq \frac{d_{min}}{1+\epsilon} \), we consider next an arbitrary \( \tau \) with \( d_j = d_{min} = \frac{V_{new}(d_{min}) + V_A}{f(m, \epsilon)} < \tau < d_{min} \cdot \left( \frac{1+\epsilon}{\epsilon} \right)^\frac{1}{h} \). Due
to Line 7 of Algorithm 1 we have
\[ V_{\min}(\tau) < V_{\min}(d_{\min}) + (\tau - d_{\min}) \cdot f(m, \varepsilon) \]
\[ \leq d_{\min} \cdot f(m, \varepsilon) \cdot \left( \frac{1 + \varepsilon}{\varepsilon} \right)^{\frac{\tau}{1 + \varepsilon}} - V_{\Delta} + + (\tau - d_{\min}) \cdot f(m, \varepsilon) \]
\[ \leq \left( \frac{\tau}{1 + \varepsilon} \right) \cdot V_{\Delta} \]
\[ \leq \left( \frac{\tau}{1 + \varepsilon} \right) \cdot V_{\Delta} \]

Therefore, Eq. (10) remains valid after the acceptance of job \( J_j \).

Finally, we allow \( d_j > d_{\min} \) and address \( d_j - p_j \) \(< \tau \) \( d_{\new} \). Due to the range of \( \tau \), we can assume \( d_j \leq d_{\new} \). For \( V_{\min}(d_{\new}) \geq V_{\min}(d_{\min}) + (d_j - d_{\min}) \cdot f(m, \varepsilon) \), we transfer processing time in direction of earlier deadlines to obtain
\[ (V_{\min}(\tau))' = \left\{ V_{\min}(\tau) \text{ for } 0 \leq \tau < d_j, \max \{d_j \cdot f(m, \varepsilon), V_{\min}(\tau)\} \text{ for } d_j \leq \tau \right\} \]

Again we use the notation \((\ldots)\)' to indicate values after the transformation. Since we do not generate a schedule after the transformation, the missing continuity of \( (V_{\min}(\tau))' \) for \( \tau = d_j \) does not matter. The transformation leads to \( (d_{\min})' = d_j \). Due to Eq. (9), Eq. (10) remains valid after the transformation and we have already shown the validity of Eq. (10) after the acceptance of \( J_j \). This validity also holds for the original values of \( (V_{\min}(\tau))' \).

For \( V_{\min}(d_{\new}) < V_{\min}(d_{\min}) + (d_j - d_{\min}) \cdot f(m, \varepsilon) \), we determine the largest value \( \delta \in (\max \{d_{\min}, d_j - p_j\}, d_j) \) with
\[ V_{\min}(d_{\new}) + (d_j - \delta) = V_{\min}(d_{\min}) + (\delta - d_{\min}) \cdot f(m, \varepsilon) \]
\[ \delta \]
exists due to
\[ V_{\min}(d_{\new}) + p_j = (d_{\new} - d_{\min}) \cdot f(m, \varepsilon) + V_{\min}(d_{\min}) + (d_j - d_{\min}) \cdot f(m, \varepsilon) + + V_{\min}(\delta - d_{\min}) \]

We split job \( J_j \) into two jobs \( J_{j1} \) and \( J_{j2} \) with \( d_{j1} = d_j, p_{j1} = d_j - \delta, \) \( d_{j2} = \delta, \) and \( p_{j2} = p_j - (d_j - \delta) \). Both jobs observe the slack condition. We first submit and accept \( J_{j1} \). The acceptance of \( J_{j1} \) does not increase \( d_{\min} \) since \( V_{\min}(\tau) \) does not increase for any \( \tau \leq \delta \). For any \( \tau \in ( \delta, d_{\new}) \), we have
\[ V_{\min}(\tau) - V_{\min}(d_{\min}) + p_{j1} \leq V_{\min}(d_{\new}) - V_{\min}(d_{\min}) + + d_j - \delta \]
\[ \leq (\delta - d_{\min}) \cdot f(m, \varepsilon) \]
\[ \leq (\tau - d_{\min}) \cdot f(m, \varepsilon) \]

Therefore, the acceptance does not change the validity of Eq. (11). For \( J_{j2} \), the previously discussed case applies.

Due to Lemma 2 Algorithms 1 and 2 guarantee \( V_{\min}(\tau)|_{t} \leq (\tau - t) \cdot m \). Therefore, there is a valid preemptive schedule for the accepted jobs due to Theorem 1 and Algorithm 2 generates a valid schedule.

We now determine the performance of Algorithms 1 and 2. We begin with introducing a property of the generated schedule if all jobs have the same submission time.

**Lemma 3** Consider a job set \( J \) with the common submission time \( 0 \) for all jobs. If there is a valid schedule for all these jobs then the repeated execution of Algorithm 2 generates a schedule \( S \) for interval \([0, t)\) with
\[ P^*[\{0, t\}](J) - P^*[0, t)(S, J) \leq \frac{1}{4} P^*[0, t)(J) \]
for any \( t > 0 \).

**Proof** If no machine is idle in schedule \( S \) then the claim clearly holds. Therefore, we assume time \( \tau < t \) to be the first time in schedule \( S \) with at least one machine being idle. Then all jobs completing after time \( \tau \) in schedule \( S \) cannot start nor are preempted after time \( \tau \). Let \( k < m \) be the number of the jobs completing after time \( \tau \) in \( S \). Algorithm 2 produces \( P^*[\{0, t\}](S, J) \geq m \cdot \tau + k \cdot (t - \tau) \)

Since a job can contribute at most processing time \( t \) to interval \([0, t)\) in any schedule, we have
\[ P^*[\{0, t\}](J) - P^*[0, t)(S, J) \leq k \cdot \tau \]
resulting in
\[ \frac{P^*[\{0, t\}](J) - P^*[0, t)(S, J)}{P^*[0, t)(J)} \leq \frac{k \cdot \tau}{m \cdot \tau + k \cdot (t - \tau) + k \cdot \tau} \]
\[ \leq \frac{k \cdot \tau}{m \cdot \tau + k \cdot t} \leq \frac{1}{4} \]

Due to inequalities \( 0 < k < m \) and \( 0 \leq \tau < t, k = 0.5 \cdot m \) and \( \tau = 0.5 \cdot t \) produce a maximum for \( k \cdot \tau / (m \cdot \tau + k \cdot t) \). □

Only Line 3 or Line 7 of Algorithm 1 may lead to an increase of \( d_{\min} \). We say an increase of \( d_{\min} \) is a time progression increase if it occurs due to Line 3. Any other increase of \( d_{\min} \) from time \( \tau \) to time \( \tau' \) does not involve progression of time and requires \( V_{\min}(\tau') - V_{\min}(\tau) = (\tau' - \tau) \cdot f(m, \varepsilon) \).

We use this distinction of increases of \( d_{\min} \) to prove the competitive ratio for Algorithms 1 and 2.
Theorem 2 The $P_m|\epsilon,\text{online,pmtn}|\sum p_j \cdot (1-U_j)$ problem admits a deterministic online algorithm with competitive ratio at most

$$\frac{m}{f(m,\epsilon)} = m \cdot (1 + \epsilon) \cdot \left( \frac{1 + \epsilon}{\epsilon} \right)^{\frac{1}{m}} - 1.$$ 

Proof We partition schedule $S$ generated by Algorithms 1 and 2 into an alternating sequence of rejection and non-rejection intervals. A non-rejection interval starts with some $d_{\text{min}}$ value such that every increase of $d_{\text{min}}$ in this interval is a time progression increase. At its end, $d_{\text{min}}$ holds. We combine any neighboring non-rejection intervals into a single non-rejection interval. Algorithm 1 does not reject any job with either its submission time or its deadline being within this interval. An interval separating two subsequent non-rejection intervals is a rejection interval and does not contain any time progression increase of $d_{\text{min}}$.

Schedule $S$ will either start with a rejection interval or with a non-rejection interval but it will always finish with a non-rejection interval. Due to Lemma 3 we have for any non-rejection interval $[t,t')$:

$$\frac{P^*[t,t')(\mathcal{J})}{P*[t,t')(S,\mathcal{J})} \leq \frac{4}{3},$$

For any rejection interval $[t,t')$, the increase of the $d_{\text{min}}$ guarantees:

$$\frac{P^*[t,t')(\mathcal{J})}{P*[t,t')(S,\mathcal{J})} \leq \frac{(t' - t) \cdot m}{(t' - t) \cdot f(m,\epsilon)} = \frac{m}{f(m,\epsilon)}.$$

Not that for $m = 1$, Algorithms 1 and 2 guarantee the same competitive ratio $\frac{4}{3}$ as the optimal greedy algorithm for a single machine.

Lower Bounds

In this subsection, we provide a lower bound of the competitive ratio for the $P_m|\epsilon,\text{online,pmtn}|\sum p_j \cdot (1-U_j)$ problem. First we describe the general concept. Our adversary controls the job sequence $\mathcal{J}$ that only uses jobs with release date 0 and consists of up to $m + 2$ blocks. In blocks 1 to $m + 1$, the adversary submits identical jobs up to a maximum number and proceeds with the next block as soon as we have accepted a block specific target number of these jobs. If we do not accept this target number then the adversary stops after submitting the above mentioned number of jobs of this block. In block $m + 2$, the adversary unconditionally submits the corresponding maximum number of jobs and stops afterwards.

The job of the first block has deadline $d_1 = 1 + \epsilon$ and an arbitrarily small positive processing time $p_1 = \delta < 1$ such that the target number $\frac{\epsilon}{\delta} \cdot \sum_{i=0}^{m-1} \left(1+\frac{1+\epsilon}{\epsilon}\right)^{\frac{1}{m}}$ is an integer. The maximum number of jobs is $\lfloor \frac{m}{1+m} \rfloor$.

The next $m$ blocks are similar. The job of block $k$ with $2 \leq k \leq m + 1$ has a tight slack with processing time $p_k = \left(1 + \frac{1+\epsilon}{\epsilon}\right)^{\frac{1}{m}}$. For these blocks, the target number is 1 while the maximum number is $\lfloor m \cdot (1+\epsilon) \rfloor$.

The job of the final block $m + 2$ has a tight slack with processing time $p_{m+2} = \frac{1+\epsilon}{\epsilon} \cdot (1 - \delta)$. The maximum number is $\lfloor m \cdot (1+\epsilon) \rfloor$ as well.

Assume that we have accepted a total processing time of at most $P_{\text{acc}}$ and that we can arrange the submitted jobs to obtain a schedule with a total processing time of at least $P_{\text{ideal}} - \text{const} \cdot \delta$ then a lower bound of the competitive ratio for this scenario is $P_{\text{ideal}}/P_{\text{acc}}$ since $\delta$ is arbitrarily small.

Theorem 3 The competitive ratio for any deterministic online algorithm for the $P_m|\epsilon,\text{online,pmtn}|\sum p_j \cdot (1-U_j)$ problem is at least

$$\left\lfloor \frac{m \cdot (1+\epsilon)}{\epsilon} \cdot \sum_{i=0}^{m-1} \left(1+\frac{1+\epsilon}{\epsilon}\right)^{\frac{1}{m}} \right\rfloor - 1.$$ 

Proof If the adversary stops before starting block 2 then the submitted total processing time is more than $m \cdot (1+\epsilon) - \delta$ while the algorithm has accepted a total processing time of at most $\epsilon \sum_{i=0}^{m-1} \left(1+\frac{1+\epsilon}{\epsilon}\right)^{\frac{1}{m}} - \delta$ resulting in a lower bound of

$$\frac{m \cdot (1+\epsilon)}{\epsilon} \cdot \sum_{i=0}^{m-1} \left(1+\frac{1+\epsilon}{\epsilon}\right)^{\frac{1}{m}} - 1.$$ 

If the adversary stops before starting block $k + 1$ with $2 \leq k \leq m + 1$ then the algorithm has accepted a total processing time of $p_k \cdot \epsilon \sum_{i=0}^{m-1} \left(1+\frac{1+\epsilon}{\epsilon}\right)^{\frac{1}{m}}$ due to $p_2 = 1$ and

$$P_h = p_h \cdot \epsilon \sum_{i=0}^{m-1} \left(1+\frac{1+\epsilon}{\epsilon}\right)^{\frac{1}{m}} + p_h$$

$$= p_h \cdot \epsilon \cdot \left( \sum_{i=0}^{m-1} \left(1+\frac{1+\epsilon}{\epsilon}\right)^{\frac{1}{m}} + 1 \right)$$

$$= p_h \left(1+\frac{1+\epsilon}{\epsilon}\right)^{\frac{1}{m}} \cdot \epsilon \sum_{i=0}^{m-1} \left(1+\frac{1+\epsilon}{\epsilon}\right)^{\frac{1}{m}}$$

$$= p_h \left(1+\frac{1+\epsilon}{\epsilon}\right)^{\frac{1}{m}} \cdot \epsilon \sum_{i=0}^{m-1} \left(1+\frac{1+\epsilon}{\epsilon}\right)^{\frac{1}{m}}$$

for $2 \leq h < k$. 

$$= p_h+1 \cdot \epsilon \sum_{i=0}^{m-1} \left(1+\frac{1+\epsilon}{\epsilon}\right)^{\frac{1}{m}}$$
Due to Theorem 1, we can use at least all $[m \cdot (1 + \varepsilon)]$ jobs with processing time $p_k$ in the optimal schedule. This usage results in a lower bound of

$$
\frac{p_k \cdot [m \cdot (1 + \varepsilon)]}{p_k \cdot \varepsilon \cdot \sum_{i=0}^{m-1} \frac{1}{\varepsilon} \cdot \frac{1}{\varepsilon} \cdot \varepsilon^i} = \frac{[m \cdot (1 + \varepsilon)]}{\varepsilon \cdot \sum_{i=0}^{m-1} \frac{1}{\varepsilon} \cdot \varepsilon^i} = [m \cdot (1 + \varepsilon)] \cdot \left( \frac{1 + \varepsilon}{\varepsilon} \right)^{-1} - 1.
$$

If the adversary starts with block $m + 2$ then the algorithm has accepted exactly one job with deadline $d_k$ for $3 \leq k \leq m + 1$ while all other jobs have deadline $d_1 = d_2$. The total accepted processing time is

$$
P_{\text{acc}} = p_{m+1} \cdot \varepsilon \cdot \sum_{i=0}^{m-1} \left( \frac{1 + \varepsilon}{\varepsilon} \right)^{\frac{i}{m}} + p_{m+1}
$$

resulting in no idle time on any machine in interval $[0, d_1]$. Since any job of block $m + 2$ cannot start later than time

$$
d_{m+2} - p_{m+2} = \varepsilon \cdot p_{m+2} = \varepsilon \cdot \left( \frac{1 + \varepsilon}{\varepsilon}, (1 - \delta) \right) = (1 + \varepsilon) \cdot (1 - \delta) < d_1,
$$

we cannot accept any job of block $m + 2$.

Again, the total processing time of the optimal schedule is at least $[m \cdot (1 + \varepsilon)] \cdot p_{m+2}$ resulting in a lower bound of

$$
\frac{p_{m+2} \cdot [m \cdot (1 + \varepsilon)]}{(1 + \varepsilon) \cdot \sum_{i=1}^{m} \frac{1}{\varepsilon} \cdot \frac{1}{\varepsilon} \cdot \varepsilon^i} = \frac{[m \cdot (1 + \varepsilon)]}{\varepsilon \cdot \sum_{i=0}^{m-1} \frac{1}{\varepsilon} \cdot \varepsilon^i} = [m \cdot (1 + \varepsilon)] \cdot \left( \frac{1 + \varepsilon}{\varepsilon} \right)^{-1} - 1.
$$

Theorems 3 and 2 show that Algorithms 1 and 2 guarantee a tight (for $m \cdot (1 + \varepsilon)$ being an integer) or an almost tight competitive ratio for $\varepsilon \leq 1$.

We can increase the lower bound to

$$
\max \left\{ m \cdot (1 + \varepsilon), [m \cdot (1 + \varepsilon)] + \varepsilon \cdot \frac{1}{1 + \varepsilon} \cdot \left( \frac{1 + \varepsilon}{\varepsilon} \right)^{\frac{1}{m}} \right\}
$$

by adding accepted jobs with deadline $d_1$ in the optimal schedule if there are idle machines. The new lower bound reduces the gap for non-integral values of $m \cdot (1 + \varepsilon)$ but cannot guarantee a tight lower bound for all those values.

Algorithms 1 and 2 guarantee a better competitive ratio than the known greedy approach for all values of $\varepsilon$ and $m > 1$. In comparison to the single machine problem, the competitive factor is significantly better for small values of $\varepsilon$. Although we focus on small values of $\varepsilon$, we like to remark that for the parallel problem, the competitive ratio cannot approach 1 for large values of $\varepsilon$ due to the scheduling of the jobs (see Algorithm 2) while the competitive ratio tends to 1 for $\varepsilon \to \infty$ in the single machine problem.

### 4 Online Scheduling without Preemption

In this section, we present an online algorithm for the non-preemptive version. Our non-preemptive Algorithm 3 Online Allocation also uses a deadline threshold and determines this threshold based on an exponential sequence of processing times on the machines. Remember that in our online model, the algorithm must decide immediately after submission of a job whether to accept or reject the job. If we accept the job then we also must immediately allocate the job to a specific machine with a specific starting time and cannot change this allocation later. Once we have allocated a job to a machine, we start this job as early as possible on this machine. Therefore, it is not necessary to consider the job schedule but only the not yet completed load $l(m_i)$ on each machine $m_i$. We index the machines in decreasing order of their loads: $l(m_1) \geq l(m_2) \geq \ldots \geq l(m_{m-1}) \geq l(m_m)$. We use the notation $l(m_i)|_t$ to describe the value $l(m_i)$ at a specific time $t$ although we are aware that there may be different load values at time $t$ due to the acceptance of jobs. For disambiguation, we apply a similar notation $l(m_i)|_{t_j}$ to indicate the value $l(m_i)$ when making the decision to accept job $J_i$. $l(m_i)|_{t_j}$ describes the update of the load $l(m_i)$ after accepting job $J_i$. Note that the allocation of $J_i$ to a machine may result in a re-indexing of the machines. This re-indexing is already considered in $l(m_i)|_{t_j}$.

The next two expressions specify our deadline threshold:

$$
\lim_{t \to 0} d_{\text{lim}} = l(m_i)|_t \left( \frac{1 + \varepsilon}{\varepsilon} \right)^{\frac{1}{m}} + t
$$

$$
\lim_{t \to 0} d_{\text{lim}} = \max_{1 \leq i \leq m} d_{\text{lim}}|_{t_j}
$$

If $d_{\text{lim}}|_{t_j} = d_{\text{lim}}|_{t_j}$ holds then we say that machine $m_i$ determines $d_{\text{lim}}$. We use the notation $d_{\text{lim}}|_{t_j}$ to denote the value of $d_{\text{lim}}|_{t_j}$ after the acceptance of job $J_i$.

In Algorithm 3 Line 1, we initially set $d_{\text{lim}}$ to 0. At the submission of a new job $J_i$, we first update $d_{\text{lim}}$ to consider any progression of time since the previous submission (Line
3). If the resulting $d_{\text{lim}}|J_t > d_t$ holds then we reject $J_t$. Otherwise we accept $J_t$ and allocate it to the machine that produces the smallest value $d_{\text{lim}}|J_t$ (Lines 7 to 9). Contrary to Algorithm 4, Algorithm 5 allows a decrease of $d_{\text{lim}}$ due to progression of time. Therefore, there is no need for any compensation value.

Algorithm 3: Online Allocation

1: $d_{\text{lim}} = 0$
2: for each newly submitted job $J_t$ do
3:   update $d_{\text{lim}}$
4:   if $d_t < d_{\text{lim}}$ then
5:     reject $J_t$
6:   else
7:     accept $J_t$
8:     allocate $J_t$ to a machine producing minimal $d_{\text{lim}}$
9:   update $d_{\text{lim}}$
10: end if
11: end for

Lemma 4 There is a legal schedule for all jobs accepted by Algorithm 3.

Proof Acceptance of job $J_t$ requires $d_t \geq d_{\text{lim}}|J_t$. If $J_t$ is allocated to machine $m_i$ then it completes at

$$l(m_i)|J_t + r_j + p_j \leq (d_{\text{lim}}|J_t - r_j) + \frac{\epsilon}{1 + \epsilon} + r_j + p_j \leq (d_t - r_j) + \frac{\epsilon}{1 + \epsilon} + r_j + \frac{d_t - r_j}{1 + \epsilon} \leq d_t.$$  

For the proof of the competitive factor, we first establish a property of the schedule.

Lemma 5 Algorithm 3 always guarantees

$$l(m_1)|J_t + l(m_2)|J_t \geq (d_{\text{lim}}|L - t) \cdot \left( \frac{\epsilon}{1 + \epsilon} \right)^{\frac{1}{n}}.$$  

Proof The claim holds whenever there is no load on any machine or if machine $m_1$ determines $d_{\text{lim}}$ due to Eq. (19).

Next, we assume that Algorithm 3 accepts job $J_t$ and that the claim holds before accepting $J_t$. Algorithm 3 always guarantees $d_{\text{lim}}|J_t < d_t$. We have already stated that the claim holds in the latter case while in the former case, the claim continues to hold since
no machine load decreases. Therefore, we assume \( k > 1 \) and \( d_{lim,j}' < d_{lim,j} \), that is, machine \( m_h \) with \( h > 1 \) determines \( d_{lim,j}' \).

For \( l(m_1)|j_j + p_j + r_j \leq d_j \), Line 8 of Algorithm 3 guarantees

\[
l(m_1)|j_j + l(m_2)|j_j \geq l(m_1)|j_j + p_j
\]

\[
> (d_{lim,j}' - r_j) \cdot \left( \frac{1 + \epsilon}{1 + \frac{1}{k}} \right)^{\frac{1}{k}}.
\]

For \( l(m_1)|j_j + p_j + r_j > d_j \geq d_{lim,j} \), we have either \( d_{lim,j} = d_{lim,j}' \) and the claim continues to hold or \( i \leq h \leq k \) with \( l(m_1)|j_j \leq l(m_{h-1})|j_j \) and

\[
d_{lim,j}' - r_j = d_{lim,j}' - r_j = \frac{(1 + \epsilon)}{1 + \frac{1}{k}} \cdot \left( \frac{1 + \epsilon}{1 + \frac{1}{k}} \right)^{\frac{1}{k}}
\]

\[
\leq l(m_{h-1})|j_j \cdot \left( \frac{1 + \epsilon}{1 + \frac{1}{k}} \right)^{\frac{1}{k}} \cdot \left( \frac{1 + \epsilon}{1 + \frac{1}{k}} \right)^{\frac{1}{k}}
\]

\[
\leq l(m_{h-1})|j_j \cdot \left( \frac{1 + \epsilon}{1 + \frac{1}{k}} \right)^{\frac{1}{k}}
\]

\[
\leq l(m_1)|j_j + l(m_2)|j_j \cdot \left( \frac{1 + \epsilon}{1 + \frac{1}{k}} \right)^{\frac{1}{k}}.
\]

Note that any time instance of schedule \( S \) with at least one busy machine belongs to a busy interval and it is not possible to execute a rejected job at a time instance that does not belong to any usable interval. Such a restriction does not exist for an accepted job.

**Theorem 4** The \( P_{m|\epsilon,online} | \sum p_j \cdot (1 - U_j) \) problem admits a deterministic online algorithm with competitive ratio at most

\[
m \cdot \left( \frac{1 + \epsilon}{\epsilon} \right)^{\frac{1}{k}} + 1.
\]

**Proof** Since no two busy intervals overlap, we assume \( S \) to be a single busy interval \( [t, t'] \) with the corresponding usable interval \( [t, t'' \leq \max_{j \in \mathcal{J}} \{d_j \}] \). Then we have

\[
P^*(0, t'') \leq P^*(0, t') \cdot \left( \frac{1 + \epsilon}{\epsilon} \right)^{\frac{1}{k}} + 1
\]

\[
\leq m \cdot P^*(0, t') \cdot \left( \frac{1 + \epsilon}{\epsilon} \right)^{\frac{1}{k}} + 1
\]

\[
\leq m \cdot \left( \frac{1 + \epsilon}{\epsilon} \right)^{\frac{1}{k}} + 1.
\]

The expression \( m \cdot \left( \frac{1 + \epsilon}{\epsilon} \right)^{\frac{1}{k}} + 1 \) has its minimal value \( e \cdot \ln \frac{1 + \epsilon}{\epsilon} + 1 \) for \( m = \ln \frac{1 + \epsilon}{\epsilon} \). Since it decreases for \( 1 \leq m \leq \ln \frac{1 + \epsilon}{\epsilon} \) and increases for \( m \geq \ln \frac{1 + \epsilon}{\epsilon} \), we may obtain a better competitive ratio by partitioning the system in groups of approximately \( \ln \frac{1 + \epsilon}{\epsilon} \) machines. In particular, this approach yields the following result.

**Corollary 1** For \( \ln \frac{1 + \epsilon}{\epsilon} \) being an integer and \( m = i \cdot \ln \frac{1 + \epsilon}{\epsilon}, \ i \in \mathbb{N}, \ \text{the} \ P_{m|\epsilon,online} | \sum p_j \cdot (1 - U_j) \) problem admits a deterministic online algorithm with competitive ratio at most

\[
e \cdot \ln \left( \frac{1 + \epsilon}{\epsilon} \right) + 1.
\]

**Proof** We partition the machines into sets of machines with size \( \ln \frac{1 + \epsilon}{\epsilon} \). For each set of machines, we run Algorithm 3 and obtain a competitive ratio of at most \( e \cdot \ln \frac{1 + \epsilon}{\epsilon} + 1 \) for the set of jobs allocated to these machines. We consider any job not allocated to these machines on the next set of machines and repeat the procedure. Hence the competitive ratio of Theorem 4 with \( m = \ln \frac{1 + \epsilon}{\epsilon} \) carries over.

For \( m = 1 \), Theorem 4 produces the competitive ratio of the single machine case although Algorithm 3 does not greedily accept every job that we can schedule: in particular, it rejects jobs with a deadline below the threshold deadline even if their small processing time allows a valid schedule. This rejection has no influence on the competitive ratio.
For $m > 1$ and $\frac{\ln m}{m} \leq \varepsilon \Leftrightarrow \varepsilon > 0.58$, greedy acceptance has a better competitive ratio than Algorithm 3 while the competitive ratio of Algorithm 3 is clearly better for smaller values of $\varepsilon$ and $m > 1$.

We also briefly point out how to use our deterministic algorithm for parallel machines to obtain a randomized algorithm for small values of $\varepsilon$.

**Corollary 2** The $1|\varepsilon,\text{online}|\sum p_j \cdot (1 - U_j)$ problem admits a randomized online algorithm against an oblivious adversary with competitive ratio at most

$$\min_{m \in \{\lfloor \ln (1+\varepsilon) \rfloor, \lfloor \ln (1/\varepsilon) \rfloor \}} \left\{ m^2 \cdot \left( \frac{1 + \varepsilon}{\varepsilon} \right)^{1/2} + m \right\}$$

if $\varepsilon > 0$ is sufficiently small.

**Proof** We simulate our deterministic Algorithm 3 with $m$ machines. Denote by $\text{ALG}_m$ and $\text{OPT}_m$ the obtained utilization and the optimal utilization on these machines, respectively. Then we have

$$m \cdot \left( \frac{1 + \varepsilon}{\varepsilon} \right)^{1/2} + 1 \geq \frac{\text{OPT}_m}{\text{ALG}_m} \geq \frac{\text{OPT}}{\text{ALG}}.$$

We now choose one of the simulated machines uniformly at random which, on expectation, has exactly an $\varepsilon$-competitive ratio. Then the adversary continues with the submission of a job of the final group. Finally the adversary submits $m$ jobs with submission time $t$ and processing time $\frac{1}{\varepsilon} - \delta$ for an arbitrary small $\delta > 0$. As in the approach of Theorem 3 we are not able to schedule any job of the final group.

**Theorem 5** For the $P_m|\varepsilon,\text{online}|\sum p_j \cdot (1 - U_j)$ problem with $\varepsilon < 1$, any deterministic online algorithm is at least strictly $c$-competitive with

$$\frac{c}{m} = \left( \frac{m}{(c-1)} \right)^{1/\varepsilon} - 1.$$ 

**Proof** For $\varepsilon < 1$, we cannot allocate any two jobs $J_i$ and $J_k$ to the same machine if both jobs have a tight slack, $r_j = r_k = t$ and $1 \leq p_j, p_k < \frac{1}{\varepsilon}$. Since job $J_1$ starts at time $t$, we can include it into this group of jobs, that is, we cannot allocate any other job of this group to the same machine as job 1.

Assume that we want to test whether a competitive ratio $c$ is a strict lower bound. Then we must show that there is a set of increasing processing times $p_1 = 1 < p_2 < \ldots < p_m < p_{m+1} < \frac{1}{\varepsilon}$ such that we have

$$1 + m \cdot p_j \geq c \cdot \sum_{i=1}^{j-1} p_i \quad \text{with} \quad 1 < j \leq m + 1. \quad (24)$$

Eq. (24) assumes that we have accepted exactly one job from block 1 to block $j - 1$ but no job from block $j$. As already mentioned, we can only schedule the accepted jobs using exactly $j - 1$ machines. In the optimal schedule, we execute job $J_1$ such that it either completes before $t$ or starts at $t + \frac{1}{\varepsilon}$ while we allocate $m$ jobs with processing time $p_j$ to a separate machine each starting at time $t$. Since the right hand side of Eq. (24) is a strict lower bound for the single machine preemptive problem, we have

$$c \frac{m}{(c-1)} \left( \frac{1}{\varepsilon} - \delta \right).$$

To minimize $c$, we assume a value $p_2$ with $1 < p_2 < \frac{1}{\varepsilon}$ and $p_{m+1} = \frac{1}{\varepsilon}$. We justify the second assumption by the fact that the difference between $p_{m+1}$ and $\frac{1}{\varepsilon}$ is arbitrarily
small. We transform Eq. (24) into a set of equations and then subtract the equation for \( p_j \) from the equation for \( p_{j-1} \):

\[
p_2 = \frac{c - 1}{m} \\
p_j = p_{j-1} \cdot \left( \frac{c}{m} + 1 \right) \quad \text{for} \quad 2 < j \leq m + 1 \\
p_{m+1} = \frac{1}{\epsilon}
\]

Then we must solve the equation \( p_j = \left( \frac{m}{(c-1)\epsilon} \right)^{\frac{1}{j}} - 1 \). The condition \( p_2 = \frac{c - 1}{m} \) is necessary to obtain the largest value for \( c \) and guarantee the validity of Eq. (24) for \( j = 2 \). □

For large values of \( m \), we obtain approximately \( p_j \approx \left( \frac{1}{\epsilon} \right)^{\frac{1}{j}} \). For \( m = 1 \), the lower bound of Theorem 5 becomes \( 1 + \frac{1}{m} \), that is the lower bound of the corresponding preemptive problem. Theorems 2 and 5 show that there is a gap between the lower bound of the competitive ratio and the competitive ratio of Algorithm 3. Therefore, the lower bound of Theorem 5 is not the best possible lower bound for this problem. Deriving a better lower bound that is applicable to all values of \( m \) remains a challenging open problem.

Acknowledgements This work was partially funded by ERC Advanced Grant 788893 AMDROMA.

References

1. Alon, N., Azar, Y., Berlin, M.: The price of bounded preemption. In: Proceedings of the 30th Symposium on Parallelism in Algorithms and Architectures, SPAA ‘18, pp. 301–310. ACM, New York, NY, USA (2018)
2. Awerbuch, B., Azar, Y., Fiat, A., Leonardi, S., Rosén, A.: On-line competitive algorithms for call admission in optical networks. Algorithmica 31(1), 29–43 (2001)
3. Azar, Y., Kalp-Shaltiel, I., Lucier, B., Menache, I., Naor, J., Yaniv, J.: Truthful online scheduling with commitments. In: Proceedings of the Sixteenth ACM Conference on Economics and Computation, EC ’15, Portland, OR, USA, June 15-19, 2015, pp. 715–732 (2015)
4. Baruah, S., Haritsa, J.: Scheduling for overload in real-time systems. IEEE Trans. Computers 46(9), 1034–1039 (1997)
5. Baruah, S.K., Koren, G., Mao, D., Mishra, B., Raghunathan, A., Rosier, L.E., Shasha, D., Wang, F.: On the competitiveness of online real-time task scheduling. Real-Time Systems 4(2), 125–144 (1992)
6. Bundde, D., Goldwasser, M.: Dispatching equal-length jobs to parallel machines to maximize throughput. Lecture Notes in Computer Science 6139, 346–358 (2010)
7. Canetti, R., Irani, S.: Bounding the power of preemption in randomized scheduling. SIAM J. Comput. 27(4), 993–1015 (1998)
8. Chen, L., Eberle, F., Megow, N., Schewior, K., Stein, C.: A general framework for handling commitment in online throughput maximization. CoRR abs/1811.08238 (2018). URL http://arxiv.org/abs/1811.08238
9. DasGupta, B., Palis, M.: Online real-time preemptive scheduling of jobs with deadlines on multiple machines. Journal of Scheduling 4(6), 297–312 (2001)
10. Garay, J., Naor, J., Yener, B., Zhao, P.: On-line admission control and packet scheduling with interleaving. In: Proceedings IEEE INFOCOM 2002, The 21st Annual Joint Conference of the IEEE Computer and Communications Societies, New York, USA, June 23-27, 2002, pp. 94–103 (2002)
11. Goldman, S., Parwatikar, J., Suri, S.: Online scheduling with hard deadlines. Journal of Algorithms 34(2), 370 – 389 (2000)
12. Goldwasser, M.: Patience is a virtue: The effect of slack on competitiveness for admission control. In: Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms, 17-19 January 1999, Baltimore, Maryland., pp. 396–405 (1999)
13. Goldwasser, M.: Patience is a virtue: the effect of slack on competitiveness for admission control. Journal of Scheduling 6(2), 183–211 (2003)
14. Graham, R., Lawler, E., Lenstra, J., Kan, A.R.: Optimization and approximation in deterministic sequencing and scheduling: a survey. In: P. Hammer, E. Johnson, B. Korte (eds.). Discrete Optimization II, Annals of Discrete Mathematics, vol. 5, pp. 287 – 326. Elsevier (1979)
15. Horn, W.: Some simple scheduling algorithms. Naval Research Logistics (NRL) 21(1), 177–185 (1974)
16. Kalyanasundaram, B., Pr rhs, K.: Eliminating migration in multi-processor scheduling. J. Algorithms 38(1), 2–24 (2001)
17. Kalyanasundaram, B., Pr hu s, K.: Maximizing job completions online. J. Algorithms 49(1), 63–85 (2003)
18. Kim, J., Chwa, K.Y.: On-line deadline scheduling on multiple resources. In: Computing and Combinatorics, 7th Annual International Conference, COCOON 2001, Guilin, China, August 20-23, 2001, Proceedings, pp. 443–452 (2001)
19. Koren, G., Shasha, D.: MOCA: A multiprocessor on-line competitive algorithm for real-time system scheduling. Theor. Comput. Sci. 128(1&2), 75–97 (1994)
20. Lawler, E.: A dynamic programming algorithm for preemptive scheduling of a single machine to minimize the number of late jobs. Ann. Oper. Res. 26(1-4), 125–133 (1991)
21. Lee, J.: Online deadline scheduling: multiple machines and randomization. In: SPAA 2003: Proceedings of the Fifteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures, June 7-9, 2003, San Diego, California, USA (part of FCRC 2003), pp. 19–23 (2003)
22. Lipton, R., Tomkins, A.: Online interval scheduling. In: Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, 23-25 January 1994, Arlington, Virginia., pp. 302–311 (1994)
23. Lucier, B., Menache, I., Naor, J., Yaniv, J.: Efficient online scheduling for deadline-sensitive jobs: extended abstract. In: 25th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA ’13, Montreal, QC, Canada - July 23 - 25, 2013, pp. 305–314 (2013)
24. Porter, R.: Mechanism design for online real-time scheduling. In: Proceedings 5th ACM Conference on Electronic Commerce (EC-2004), New York, NY, USA, May 17-20, 2004, pp. 61–70 (2004)
25. Prohs, K., Woeginger, G.: Approximation schemes for a class of subset selection problems. Theor. Comput. Sci. 382(2), 151–156 (2007)
26. Schwiegelshohn, C., Schwiegelshohn, U.: The power of migration for online slack scheduling. In: 24th Annual European Symposium on Algorithms, ESA 2016, August 22-24, 2016, Aarhus, Denmark, pp. 75:1–75:17 (2016)
27. Woeginger, G.: On-line scheduling of jobs with fixed start and end times. Theor. Comput. Sci. 130(1), 5–16 (1994)