The Power of Migration for Online Slack Scheduling

Chris Schwiegelshohn[*1] **and Uwe Schwiegelshohn**[2]

1 **Computer Science Department, TU Dortmund, Dortmund, Germany**
chris.schwiegelshohn@tu-dortmund.de
2 **Department of Electrical Engineering and Information Technology, TU Dortmund, Dortmund, Germany**
uwe.schwiegelshohn@tu-dortmund.de

Abstract

We investigate the power of migration in online scheduling for parallel identical machines. Our objective is to maximize the total processing time of accepted jobs. Once we decide to accept a job, we have to complete it before its deadline d that satisfies $d \geq (1+\varepsilon) \cdot p + r$, where p is the processing time, r the submission time and the slack $\varepsilon > 0$ a system parameter. Typically, the hard case arises for small slack $\varepsilon \ll 1$, i.e. for near-tight deadlines. Without migration, a greedy acceptance policy is known to be an optimal deterministic online algorithm with a competitive factor of $\frac{1+\varepsilon}{\varepsilon}$ (DasGupta and Palis, APPROX 2000). Our first contribution is to show that migrations do not improve the competitive ratio of the greedy acceptance policy, i.e. the competitive ratio remains $\frac{1+\varepsilon}{\varepsilon}$ for any number of machines.

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

1998 ACM Subject Classification F.1.2 Online Computation

Keywords and phrases Online scheduling, deadlines, preemption with migration, competitive analysis

Digital Object Identifier 10.4230/LIPIcs.ESA.2016.75

1 Introduction

We address a basic scheduling problem on parallel identical machines. Given a sequence of jobs, we are required to use the available resources as best as possible such that no accepted job exceeds its deadline. The deadline of a job is at least $(1+\varepsilon) \cdot p_j$ time units after its submission time, where p_j is the processing time of the job and $\varepsilon > 0$ is the (typically small) slack parameter. We model resource utilization by maximizing the total processing time. In addition to the slack, we require the system to support preemption and migration.

From a system's perspective, migration is an important topic. Not only is it technically feasible, at least to some degree, in some cases it is even required [2, 8, 26]. For instance, when

* The first author wants to acknowledge the German Research Council (DFG) for its support of this research within the Collaborative Research Center SFB 876, project A2.

outsourcing jobs to an external provider, these jobs are often executed via virtual machines instead of given access to a real server. Since the underlying systems are highly adaptive environments dealing with machine failure and maintenance, migration is a necessary part of such virtual infrastructure management [25].

From a theoretical perspective, assuming migration is a convenient simplification when designing algorithms. Additionally, understanding when and how much migration improves performance can ease the decision whether implementing migration in a system is worthwhile. For a few examples where migration improves the performance of online algorithms, we refer to [1, 6, 9, 22, 24].

1.1 Our Contribution

In this paper, we consider a system of m parallel identical machines supporting preemption with migration and process jobs in an online fashion. A job can be interrupted at any given time and resumed at a later date on a possibly different machine without overhead. At any given time, a job can only be executed on one machine and each machine can process at most one job. The slack $\varepsilon > 0$ is known to the algorithm a-priori and upon submission of a job J_j at time r_j, the processing time p_j and deadline $d_j \geq (1+\varepsilon) \cdot p_j + r_j$ are revealed to the algorithm. We must immediately decide whether or not to accept J_j, denoted by the indicator variable $U_j = 1$ if rejected and $U_j = 0$ if accepted and receive a payoff proportionate to p_j, if J_j is accepted. Once a job is accepted, we must complete it on time, i.e. we cannot postpone completion of a job beyond its deadline in favor of another job. Using the three-field notation, the problem is $P_m|\varepsilon, \text{online}, \text{pmtn}|\sum p_j \cdot (1-U_j)$. We measure the performance of online algorithms via the competitive ratio. Since we have a maximization problem, the competitive ratio is the maximum value of $\frac{\text{OPT}(J)}{\text{Alg}(J)}$ over all input sequences J, where $\text{OPT}(J)$ is the objective value of an optimal offline algorithm and $\text{Alg}(J)$ is the objective value achieved by our online algorithm.

To accurately present our results, we require the definition of the function $g(x) = \varepsilon \cdot \left(\frac{1+\varepsilon}{\varepsilon}\right)^x$. Our main contribution is the design of a deterministic algorithm with an almost tight competitive factor.

▶ **Theorem 1.** *The $P_m|\varepsilon, online, pmtn|\sum p_j \cdot (1-U_j)$ problem admits a deterministic online algorithm with competitive ratio at most* $\max\left\{\frac{m\cdot(1+\varepsilon)}{\sum_{i=0}^{m-1} g\left(\frac{i}{m}\right)}, \frac{4}{3}\right\}$.

We are especially interested in the behavior of the function $\frac{m\cdot(1+\varepsilon)}{\sum_{i=0}^{m-1} g\left(\frac{i}{m}\right)}$ for small ε and large m, though we note that for any value of $0 < \varepsilon \leq 1$ and $m > 1$, the competitive ratio of our algorithm is an improvement over the greedy algorithm. For a single machine, there is no migration and the expression reduces to $\frac{1+\varepsilon}{\varepsilon}$. As m tends to infinity, the expression approaches $(1+\varepsilon)\ln\frac{1+\varepsilon}{\varepsilon}$. For small ε, this is an exponential improvement over greedy. We also note that the competitive ratio improves quickly, i.e. even for a comparatively small number of machines a parallel system drastically outperforms a single machine environment. Generally, for m machines, the competitive factor of our algorithm is $O(\sqrt[m]{1/\varepsilon})$. For an arguably more immediate feel for the competitive ratios of various choices of ε and m, we refer to Figure 1.

We further prove the following lower bound.

▶ **Theorem 2.** *Any deterministic online algorithm for the $P_m|\varepsilon, online, pmtn|\sum p_j \cdot (1-U_j)$ problem has a competitive ratio of at least* $\frac{\lfloor m\cdot(1+\varepsilon)\rfloor}{\sum_{i=0}^{m-1} g\left(\frac{i}{m}\right)} \cdot (1-\delta)$ *for any $\delta > 0$.*

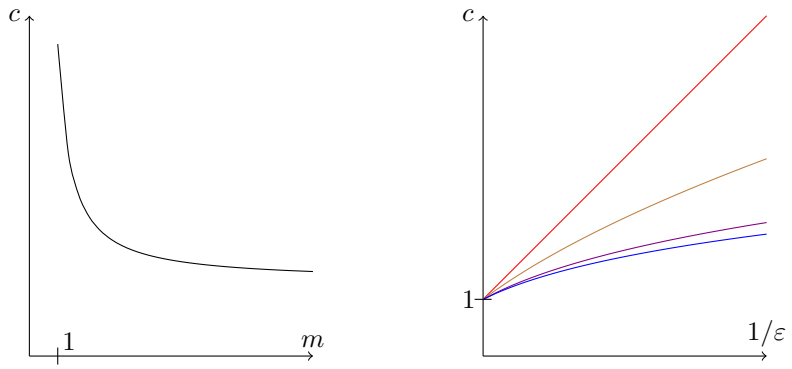

Figure 1 Performance of the online algorithm for increasing number of machines. The plots contain parameterizations of the function $\frac{m\cdot(1+\varepsilon)}{\sum_{i=0}^{m-1}\varepsilon\cdot\left(\frac{1+\varepsilon}{\varepsilon}\right)^{i/m}}$, which is the competitive ratio of our algorithm for $0 < \varepsilon \leq 1$. We have $\varepsilon = 0.1$ in the left figure and the x-axis and the y-axis denote the number of machines and the competitive ratio, respectively. In the right figure, the x-axis and y-axis denote the dependency on $1/\varepsilon$ (i.e. $1/\varepsilon$ is large when the slack ε is small) and the competitive ratio, respectively. The red line is the competitive ratio of the greedy algorithm for increasingly smaller slack; the brown, violet and blue lines are the competitive ratios of our algorithm for 2, 10 and an infinite number of machines, respectively. Note that the competitive ratio is not equivalent to the plotted function for large ε (i.e. $1/\varepsilon$ close to 0).

When $m \cdot (1 + \varepsilon)$ is integral and $c_A \geq \frac{4}{3}$, our algorithm achieves a tight competitive ratio.

The greedy acceptance policy accepts any job that can be completed on time provided no other accepted job is delayed beyond its deadline. For the same problem without migrations, this algorithm is known to guarantee an optimal competitive ratio of $\frac{1+\varepsilon}{\varepsilon}$ [7]. We show that the greedy acceptance policy cannot benefit from migrations:

▶ **Theorem 3.** *The greedy acceptance policy for the $P_m|\varepsilon, online, pmtn|\sum p_j\cdot(1-U_j)$ problem has a competitive ratio of at least $\frac{1+\varepsilon}{\varepsilon} - \delta$ for any $\delta > 0$.*

Since the non-migratory and the migratory models are identical for a single machine, the greedy algorithm is optimal in this case. But its performance is exponentially worse than the performance of our algorithm as $\varepsilon \to 0$ and $m \to \infty$.

1.2 Related Work

The offline problems $1|r_j, \text{pmtn}|\sum w_j \cdot (1 - U_j)$ and $P_2|r_j, \text{pmtn}|\sum U_j$ are NP hard [5]. Lawler [18] proposed a dynamic programming algorithm over the range of weights for the former problem, i.e. $1|r_j, \text{pmtn}|\sum(1 - U_j)$ is solvable in polynomial time. Kalyanasundaram and Pruhs [13] showed that any schedule for the $P_m|r_j, pmtn|\sum w_j \cdot (1 - U_j)$ problem can be transformed into a non-migratory schedule with $O(1)$ additional machines.

Early preemptive online algorithms assumed that every job can be accepted, but possibly delayed arbitrarily. A payoff w_j is only received if the job is completed prior to deadline. In this case there exists a tight 4-competitive deterministic algorithm on a single machine for a class of well-behaved payoff functions including $w_j = p_j$ [3, 17, 27]. For arbitrary payoff functions and without further restrictions on the input, the competitive factor is unbounded [27]. Kalyanasundaram and Pruhs [14] gave a $O(1)$ competitive randomized algorithm for maximizing the number of completed jobs and also showed that no constant competitive deterministic algorithm exists.

When further requiring that any accepted job must be completed before its deadline, there exists no constant competitive algorithm in general. To obtain more meaningful results, research incorporated the slack $\varepsilon > 0$[1]. For $1|\varepsilon, \text{online}, \text{pmtn}|\sum p_j \cdot (1 - U_j)$, Baruah and Haritsa [4] showed that a greedy acceptance policy achieves an optimal $\frac{1+\varepsilon}{\varepsilon}$ competitive ratio. This result was extended to parallel machines supporting preemption but not migration by DasGupta and Palis [7]

A lot of work has also been done for the non-preemptive variants of the problem. In this case, the difficulty of obtaining a meaningful competitive ratio was first addressed by bounding the aspect ratio $\Delta = \frac{\max p}{\min p}$. Lipton and Tomkins [20] gave a lower bound of $\Omega(\log \Delta)$ for the competitive factor of any randomized online algorithm on a single machine and additionally assuming $d_j = p_j + r_j$ for all jobs, obtained an upper bound of $O(\log^{1+\delta} \Delta)$ for any $\delta > 0$. For arbitrary deadlines, Goldman, Parwatikar, and Suri [10] were able to obtain a $6(\lceil \log \Delta \rceil + 1)$ competitive algorithm. They further studied the problem with slack ε and achieved a competitive factor of $1 + \frac{\lceil \varepsilon \rceil}{\varepsilon}$ if there are only two different job processing times. Goldwasser [11] distinguished between exactly two distinct processing times or arbitrary processing times and gave tight deterministic competitive factors of $1 + \max\left\{\frac{\lceil \varepsilon \rceil + 1}{\lceil \varepsilon \rceil}, \frac{\lfloor \varepsilon \rfloor + 1}{\varepsilon}\right\}$ and $2 + \frac{1}{\varepsilon}$, respectively. Kim and Chwa [16] later proved the same competitive factor for a greedy acceptance policy on multiple machines. Lee [19] studied small slack factors $\varepsilon \ll 1$ by way of parallel systems. For $P_m|\text{online}|\sum p_j(1-U_j)$, he gave a deterministic $m+1+m\cdot\varepsilon^{1/m}$ competitive algorithm. Hence, if $m = \log 1/\varepsilon$, the algorithm is $O(\log 1/\varepsilon)$ competitive. By simulating the parallel algorithm and picking one machine uniformly at random, Lee further obtained an expected competitive factor of $1 + 3\log 1/\varepsilon$.

1.3 Outline

In Section 3, we describe properties of schedules if all jobs are completed prior to their deadlines. In a first order, it allows us to check whether a new job can be added to the schedule without determining the whole schedule. Furthermore, using a series of transformations and employing the slack factor property for deadlines, we are able to derive a canonical schedule without improving the competitive factor. The canonical schedule contains jobs with exponentially increasing deadlines. This exponential sequence represents a tradeoff between already accepted total processing time and the potential to accept more processing time in the future. Having determined the existence of such a schedule, the competitive ratio of our acceptance procedure follows almost immediately if all jobs have the same release date, see Section 4. When jobs arrive over time, the acceptance condition is insufficient due to the existence of jobs with large processing times. However, by introducing a careful charging scheme, we can leverage processing time of accepted jobs against processing time of rejected jobs without impacting the competitive factor, provided ε is small enough. Due to space restrictions, we omit a detailed calculus. We conclude by proving our lower bounds in Section 5.

2 Notations

Formally, we have a system of m parallel identical machines to schedule the jobs of a job sequence $\mathcal{J}$. The system allows preemption with migration, that is, it can interrupt the execution of any job and immediately or later resume the execution on a possibly different

[1] In literature the stretch $f = 1 + \varepsilon$ is often used instead. The two notions are equivalent.

machine without any preemption penalty (increase in processing time or forced additional idle time on the machine). A job $J_j \in \mathcal{J}$ has processing time p_j, release date r_j and deadline $d_j \geq (1+\varepsilon)\cdot p_j + r_j$ with $\varepsilon > 0$ being the slack factor of the system. We say that a schedule for a job system is *legal* if it completes all accepted jobs before or at their respective deadlines and each machine executes at most one job at any moment. The system receives the jobs one by one in sequence order and must irrevocably and without knowledge of any later jobs decide whether it accepts the actual job or not. However, it can only accept a job if there is a legal schedule for this job and all previously accepted jobs. We use the binary variable U_j to express the decision for job J_j: $U_j = 0$ denotes acceptance of job J_j while the rejection of J_j produces $U_j = 1$. It is our goal to maximize the total processing time of all accepted jobs $(\sum p_j \cdot (1-U_j))$. Therefore, an (online) algorithm A determines $U_j(A)$ for all jobs in $\mathcal{J}$ and produces utilization $\sum p_j \cdot (1-U_j(A))$.

We evaluate our algorithm by determining bounds for the competitive factor, that is the largest ratio between the optimal objective value and the objective value produced by online algorithm A for all possible job sequences $\mathcal{J}$: A has a competitive factor c_A if $c_A \geq \frac{p^*(\mathcal{J})}{\sum_{J_j\in\mathcal{J}}(1-U_j(A))\cdot p_j)}$ holds for any job sequence $\mathcal{J}$ with $p^*(\mathcal{J}) = \max\{\sum_{J_j\in\mathcal{J}}(1-U_j)\cdot p_j\}$ being the maximum total processing time of any legal schedule for $\mathcal{J}$. Finally, c^* is a lower bound of the competitive factor if $c^* \leq c_A$ holds for any online algorithm A.

Next we introduce function $g(x) = \varepsilon \cdot \left(\frac{1+\varepsilon}{\varepsilon}\right)^x$. Straightforward calculus yields the identity:

$$\sum_{j=i}^{m+i-1} g\left(\frac{j}{m}\right) + \left(\frac{1+\varepsilon}{\varepsilon}\right)^{\frac{i}{m}} = \sum_{j=i+1}^{m+i} g\left(\frac{j}{m}\right) \tag{1}$$

We further use g to define the following threshold expression:

$$f(m,\varepsilon) = \frac{1}{\varepsilon}\cdot\sum_{i=1}^{m} g\left(\frac{i}{m}\right) = \frac{1}{1+\varepsilon}\cdot\frac{\left(\frac{1+\varepsilon}{\varepsilon}\right)^{\frac{1}{m}}}{\left(\frac{1+\varepsilon}{\varepsilon}\right)^{\frac{1}{m}}-1}. \tag{2}$$

3 Legal Schedules

$\max\left\{\max_{J_j}\{p_j\}, \frac{1}{m}\sum_{J_j} p_j\right\}$ is the optimal makespan for the problem $P_m|\text{pmtn}|C_{\max}$, see, for instance, Pinedo [21]. Therefore, there is a legal schedule on m parallel identical machines for a set of jobs $\mathcal{J}$ with a common deadline d if and only if $p_j \leq d$ for all $J_j \in \mathcal{J}$ and $\sum_{J_j\in\mathcal{J}} p_j \leq d\cdot m$ hold. We use this result and a function $V_{\min}(t)$ to derive a similar necessary and sufficient condition for the extended problem with different deadlines. For a set of jobs $\mathcal{J}$ with possibly different deadlines, $V_{\min}(t)$ is the minimum total processing time that we must execute in interval $[0,t)$ of a legal schedule for time $t > 0$:

$$V_{\min}(t) = \sum_{J_j\in\mathcal{J}} \begin{cases} 0 & \text{for} \quad d_j - p_j \geq t \\ p_j & \text{for} \quad t \geq d_j \\ p_j - d_j + t & \text{else} \end{cases} \tag{3}$$

We sort all ν different deadlines in increasing order with d_ν being the largest deadline and set $d_0 = 0$.

▶ **Lemma 4.** *There is a legal preemptive schedule for a set of jobs $\mathcal{J}$ with deadlines on m parallel identical machines if and only if*

$$p_j \leq d_j \text{ for all } J_j \in \mathcal{J} \text{ and} \tag{4}$$

$$V_{min}(t) \leq t\cdot m \text{ for all } t > 0. \tag{5}$$

Proof. Due to Equation (3), $V_{\min}(t)$ is monotonically increasing and continuous. $\frac{d}{dt}V_{\min}(t)$ is piecewise constant and can only decrease if t is a deadline. Therefore, Inequalities (5) are valid if they hold for every deadline.

only if: Clearly if we have either $V_{\min}(d_i) > m \cdot d_i$ for at least one deadline d_i or least one job $J_j \in \mathcal{J}$ with $p_j > d_j$ then there is no legal schedule for $\mathcal{J}$.

if: We assume that Inequalities (4) and (5) hold. Then we generate a schedule for each interval $[d_{i-1}, d_i)$ in a backward order starting with $[d_{\nu-1}, d_\nu)$.

To this end, we generate an LRPT (Least Remaining Processing Time first) schedule for all jobs with deadline d_i. Since LRPT generates an optimal schedule for $P|\text{pmtn}|C_{\max}$, it produces a legal schedule within interval $[0, d_i)$ for these jobs due to the validity of Inequalities (4) for all jobs with deadline d_i and Inequality (5) for deadline d_i. We denote $\Delta = d_i - d_{i-1}$ and use interval $[0, \Delta)$ of the LRPT schedule as interval $[d_{i-1}, d_i)$ of our schedule. Then we reduce the processing time of each job by its total amount of processing within this interval and assign the new deadline d_{i-1} to all jobs with deadline d_i.

Now we must show that Inequalities (4) and (5) hold for the new largest deadline d_{i-1}. Since the LRPT schedule is legal, the total processing of a job in interval $[\Delta, d_i)$ of the LRPT schedule cannot exceed d_{i-1}. Therefore, Inequalities (4) hold for all these jobs after the modifications while Inequalities (4) remain valid for all unmodified jobs.

If there is some idleness in interval $[0, \Delta)$ of the LRPT schedule then we process $\min\{\Delta, p_j\}$ of each job J_j with deadline d_i within this interval, that is, the total processing in this interval is identical to $V_{\min}(d_i) - V_{\min}(d_{i-1})$. Therefore, $V_{\min}(d_{i-1})$ remains unchanged.

If no machine is idle in interval $[0, \Delta)$ of the LRPT schedule then $V_{\min}(d_{i-1})$ may increase and we have

$$V_{\min}(d_{i-1}) \quad = \quad V_{\min}(d_i) - m \cdot \Delta \leq m \cdot d_i - m \cdot \Delta = m \cdot d_i - m \cdot (d_i - d_{i-1}) = m \cdot d_{i-1}.$$

In both cases Inequality (5) is valid for deadline d_{i-1}. ◀

Since Lemma 4 does not consider the slack we introduce another function that leads to a sufficient but not necessary condition. First we say that a job J_j is large if $p_j > \frac{d_j}{1+\varepsilon}$ holds. Although there is no submission of large jobs, the progression of time may turn jobs into large jobs. Therefore, we must consider large jobs as well and define for every time $t > 0$:

$$V_{\text{accept}}(t) \quad = \quad \sum_{J_j \in \mathcal{J}} \begin{cases} p_j & \text{for } d_j \leq t \\ \max\{p_j - \frac{d_j}{1+\varepsilon}, 0\} & \text{for } d_j > t > d_j - \frac{d_j}{1+\varepsilon} \\ p_j - d_j + t & \text{for } d_j - \frac{d_j}{1+\varepsilon} \geq t > d_j - p_j \\ 0 & \text{for } d_j - p_j \geq t \end{cases} \tag{6}$$

Informally, we use $V_{\text{accept}}(t)$ to define a reduced threshold for accepting a new job J_j, that is, we reject a new job if for at least one time $t \geq d_j$, the new jobs leads to $V_{\text{accept}}(t) > t \cdot f(m, \varepsilon)$ - see Equation (2) - even if the acceptance of the job may produce a legal schedule.

Before formally describing this algorithm we must show that the validity of

$$V_{\text{accept}}(t) \quad \leq \quad t \cdot f(m, \varepsilon) \text{ for all } t > 0 \tag{7}$$

always guarantees a legal schedule. Therefore, the next lemma is the key lemma for our algorithm. Its proof is based on a reduction of the instance space with the help of several transformations until we obtain an instance space that we can analyze with the help of a few equations. Schwiegelshohn [23] used a similar approach to generate an alternative proof for

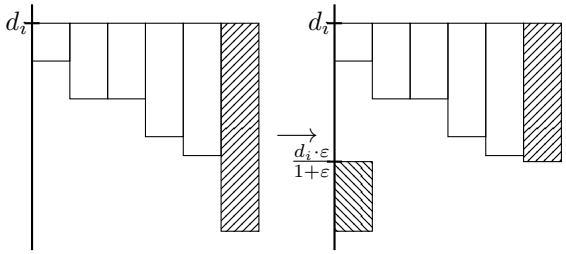

Figure 2 Large Job Splitting. The large job is marked as hatched area.

approximation factor of the lrf (largest ratio first) algorithm for the $P||\sum w_j C_j$ problem, see Kawaguchi and Kyan [15].

▶ **Lemma 5.** *There is a legal preemptive schedule for a set of jobs $\mathcal{J}$ with possibly different deadlines on m parallel identical machines if Inequalities (4) and (7) hold.*

Proof. As with function $V_{\min}(t)$ we need not examine all t to check the validity of Inequalities (7). Due to Equation (6), $V_{\text{accept}}(t)$ has the following properties:

- It is monotonically increasing.
- It is continuous unless t is a deadline.
- $\frac{d}{dt}V_{\text{accept}}(t)$ is piecewise constant.
- $\frac{d}{dt}V_{\text{accept}}(t)$ can only decrease if $t = \frac{d_j}{1+\varepsilon}$ for a large job J_j.

Therefore, Inequalities (7) are valid if they hold for every deadline d_i and the corresponding time instance $\frac{d_i}{1+\varepsilon}$ if there is a large job with deadline d_i.

We use contradiction to prove this lemma and assume that Inequalities (4) and (7) hold while there is no legal schedule for $\mathcal{J}$, that is, there is at least one deadline d_h with $V_{\min}(d_h) > d_h \cdot m$, see Lemma 4. Let d_j be the largest deadline with $V_{\min}(d_j) > d_j \cdot m$. We apply several transformations that do not decrease $V_{\min}(d_j)$ while the validity of Inequalities (4) and (7) is maintained. We use the notation $d_{>i}$ and $d_{i>}$ to describe the next larger and the next smaller deadline of deadline d_i, respectively.

First, we introduce and discuss our transformations.

Large job splitting. We split a large job J_i into one job J_{i_1} with deadline $d_{i_1} = d_i$ and processing time $p_{i_1} = \frac{d_i}{1+\varepsilon}$, and one job J_{i_2} with deadline $d_{i_2} = d_i \cdot \frac{\varepsilon}{1+\varepsilon}$ and processing time $p_{i_2} = p_i - \frac{d_i}{1+\varepsilon} \le d_i - \frac{d_i}{f} = d_{i_2}$, see Fig. 2.
This transformation neither changes $V_{\min}(t)$ for any t nor $V_{\text{accept}}(t)$ for any $t \ge d_{i_2}$. Further, it may decrease but cannot increase $V_{\text{accept}}(t)$ for any $t < d_{i_2}$. Therefore, we assume that our job sequence does not contain any large job.

Job removal. We remove any job J_i with $d_i - p_i \ge d_j$. This transformation does not change $V_{\min}(t)$ for $t \le d_j$ and cannot increase $V_{\text{accept}}(t)$ for any $t > 0$. Since other transformations may generate jobs with $d_i - p_i \ge d_j$ we must apply this transformation repeatedly.

Spread generation. Assume two jobs J_{i_1} and J_{i_2} with

$$d_{i_1} \quad = \quad d_{i_2} > d_j > d_{i_1} - p_{i_2} \ge d_{i_1} - p_{i_1} \ge d_{i_1} - \frac{d_{i_1}}{f}.$$

For job J_{i_2} we reduce its processing time p_{i_2} and its deadline d_{i_2} by $\delta < d_{i_1} - d_{i_1>}$, see Fig. 3. Due to $d_j - d_{i_1} + p_{i_2} = d_j - (d_{i_1} - \delta) + (p_{i_2} - \delta)$ this transformation does not change $V_{\min}(d_j)$ while it does not increase $V_{\text{accept}}(t)$ for $t \ne d_{i_1} - \delta$. Clearly, the

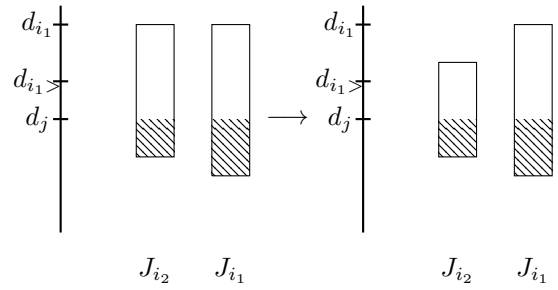

Figure 3 Spread Generation. Hatched areas contribute to $V_{\min}(d_j)$.

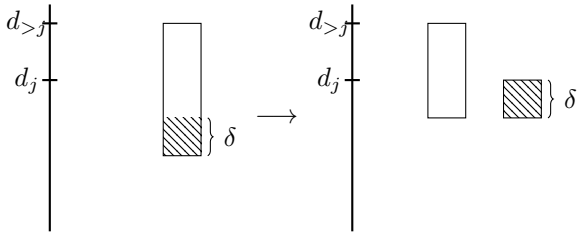

Figure 4 V_0-transformation. The hatched area becomes a new job and continues to contribute to $V_{\min}(d_j)$.

modified job J_{i_2} is not large, and we have

$$\begin{aligned} V_{\text{accept}}(d_{i_1} - \delta) &= V_{\text{accept}}(d_{i_1}) - p_{i_1} - \delta \\ &\leq d_{i_1} \cdot f(m,\varepsilon) - p_{i_1} - \delta \leq (d_{i_1} - \delta) \cdot f(m,\varepsilon) \end{aligned}$$

for $\delta \leq \frac{p_{i_1}}{f(m,\varepsilon)-1}$. Therefore, we can use any δ with $0 < \delta < \min\{d_{i_1} - d_{i_1>}, \frac{p_{i_1}}{f(m,\varepsilon)-1}\}$.

V_0-transformation. Assume $V_{\text{accept}}(d_j) = d_j \cdot f(m,\varepsilon) - \delta$ with $\delta > 0$ and a job J_i with $d_i = d_{>j}$ and $d_j > d_i - p_i$. We introduce a new job with deadline d_j and processing time $p' = \min\{\delta, d_j - d_i + p_i\}$. Note that the new job is not large since job J_i is not large. Then we reduce processing time p_i by p', see Fig. 4. This transformation does not change $V_{\min}(d_j)$ and $V_{\text{accept}}(t)$ for $t < d_j$ and $t \geq d_{>j}$. For $d_{>j} > t \geq d_j$ we increase $V_{\text{accept}}(t)$ to $d_j \cdot f(m,\varepsilon) - \delta + p' \leq d_j \cdot f(m,\varepsilon)$.

V-transformation. Assume a job J_i with $V_{\text{accept}}(d_i) = d_i \cdot f(m,\varepsilon) - \delta_V < d_i \cdot f(m,\varepsilon)$ and $V_{\text{accept}}(d_{i>}) = d_{i>} \cdot f(m,\varepsilon)$ for $d_{i>} \geq d_j$: We reduce processing time p_i of job J_i by $p' = \frac{\delta_V}{(1+\varepsilon)\cdot f(m,\varepsilon)-1}$ to p'_i and its deadline d_i by $(1+\varepsilon)\cdot p'$ to d'_i, respectively, see Fig. 5. This transformation produces $p'_i = p_i - p' \leq \frac{d_i}{1+\varepsilon} - \frac{d_i - d'_i}{1+\varepsilon} = \frac{d'_i}{1+\varepsilon}$ and

$$\begin{aligned} V_{\text{accept}}(t) &= V_{\text{accept}}(d_i) - p' = d_i \cdot f(m,\varepsilon) - \delta_V - \frac{\delta_V}{(1+\varepsilon)\cdot f(m,\varepsilon) - 1} \\ &= d_i \cdot f(m,\varepsilon) - \frac{(1+\varepsilon)\cdot\delta_V \cdot f(m,\varepsilon)}{(1+\varepsilon)\cdot f(m,\varepsilon) - 1} = (d_i - (1+\varepsilon)\cdot p') \cdot f(m,\varepsilon) \end{aligned}$$

for $d_i > t \geq d'_i$. It increases $V_{\min}(d_j)$ by $\varepsilon \cdot p'$ and decreases $V_{\text{accept}}(t)$ by p' for $t \geq d_i$. $V_{\text{accept}}(t)$ remains unchanged for $t < d'_i$. We have $d'_i > d_{i>}$ since $V_{\text{accept}}(d_i) - p_i \geq V_{\text{accept}}(d_{i>})$ holds. We apply this transformation in the order of increasing deadlines.

p-transformation. Assume two jobs J_{i_1} and J_{i_2} with $d_{i_1} > d_{i_2} > d_j > d_{i_1} - p_{i_1}$, $p_{i_1} < \frac{d_{i_1}}{1+\varepsilon}$, and $p_{i_2} < \frac{d_{i_2}}{1+\varepsilon}$. We reduce p_{i_2} by $p' = \min\{d_j - d_{i_2} + p_{i_2}, \frac{d_{i_1}}{1+\varepsilon} - p_{i_1}\}$ and increase p_{i_1} by p', see Fig. 6. This transformation does change $V_{\min}(d_j)$ and does not increase $V_{\text{accept}}(t)$ for any t.

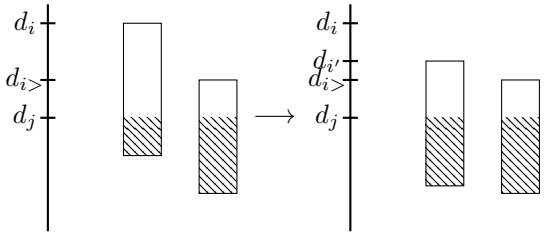

Figure 5 V-transformation. Hatched areas contribute to $V_{\min}(d_j)$.

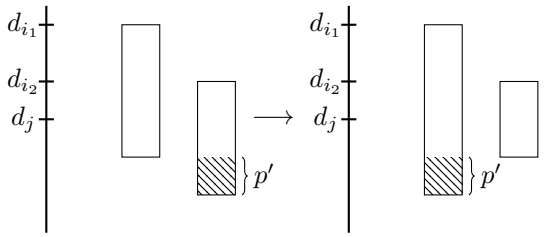

Figure 6 p-transformation. The hatched area moves to another job and continues to contribute to $V_{\min}(d_j)$.

After repeatedly applying these transformations we obtain a job sequence with the following properties:

1. $V_{\text{accept}}(d_i) = d_i \cdot f(m,\varepsilon)$ for $d_i \geq d_j$
2. $p_i < \frac{d_i}{1+\varepsilon}$ for at most one deadline $d_i > d_j$
3. $p_i = \frac{d_i}{1+\varepsilon}$ for every other deadline $d_i > d_j$
4. There are no jobs with $d_i - p_i \geq d_j$.
5. There are no two jobs with the same deadline $d_i > d_j$.

We determine the value of $V_{\text{accept}}(d_i)$ for a job J_i with $d_i > d_j$:

$$V_{\text{accept}}(d_i) \quad = \quad d_i \cdot f(m,\varepsilon) = d_i \cdot \frac{\varepsilon}{1+\varepsilon} \cdot \sum_{z=1}^{m} \left(\frac{1+\varepsilon}{\varepsilon}\right)^{\frac{z}{m}} = d_i + d_i \cdot \frac{\varepsilon}{1+\varepsilon} \cdot \sum_{z=1}^{m-1} \left(\frac{1+\varepsilon}{\varepsilon}\right)^{\frac{z}{m}}$$

For $p_i = \frac{d_i}{1+\varepsilon}$ we have

$$V_{\text{accept}}(d_i) \quad = \quad V_{\text{accept}}(d_{i>}) + p_i = d_{i>} \cdot \frac{\varepsilon}{1+\varepsilon} \cdot \sum_{z=1}^{m} \left(\frac{1+\varepsilon}{\varepsilon}\right)^{\frac{z}{m}} + \frac{d_i}{1+\varepsilon}.$$

Combining these two equations yields $d_i = d_{i>} \cdot \left(\frac{1+\varepsilon}{\varepsilon}\right)^{\frac{1}{m}}$.

First assume that $p_i = \frac{d_i}{1+\varepsilon}$ holds for every deadline $d_i > d_j$. Then we have a geometric sequence of deadlines $d_j, d_j \cdot \left(\frac{1+\varepsilon}{\varepsilon}\right)^{\frac{1}{m}}, d_j \cdot \left(\frac{1+\varepsilon}{\varepsilon}\right)^{\frac{2}{m}}, \ldots$. In this sequence, the contribution $v_{\min}(h)$ of the job with deadline $d_j \cdot \left(\frac{1+\varepsilon}{\varepsilon}\right)^{\frac{h}{m}}$ to $V_{\min}(d_j)$ is:

$$v_{\min}(h) \quad = \quad d_j - d_j \cdot \left(\frac{1+\varepsilon}{\varepsilon}\right)^{\frac{h}{m}} \cdot \left(1 - \frac{1}{1+\varepsilon}\right) = d_j \cdot \left(1 - \frac{\varepsilon}{1+\varepsilon} \cdot \left(\frac{1+\varepsilon}{\varepsilon}\right)^{\frac{h}{m}}\right)$$

Therefore, our sequence ends with $d_j \cdot \left(\frac{1+\varepsilon}{\varepsilon}\right)^{\frac{m-1}{m}}$ since no job with deadline $d_j \cdot \frac{1+\varepsilon}{\varepsilon}$ or larger can influence $V_{\min}(d_j)$ unless the job is large.

Alternatively we assume a sequence that contains a job J_i with deadline $d_i = d_j \cdot \left(\frac{1+\varepsilon}{\varepsilon}\right)^{\frac{s-1}{m}} \cdot z$ for $1 \leq s \leq m$ and $1 < z < \left(\frac{1+\varepsilon}{\varepsilon}\right)^{\frac{1}{m}}$ and calculate p_i using Equation (2):

$$d_i \cdot f(m,\varepsilon) = d_j \cdot \left(\frac{1+\varepsilon}{\varepsilon}\right)^{\frac{s-1}{m}} \cdot z \cdot f(m,\varepsilon) = d_j \cdot \left(\frac{1+\varepsilon}{\varepsilon}\right)^{\frac{s-1}{m}} \cdot f(m,\varepsilon) + p_i$$

$$\Leftrightarrow p_i = d_j \cdot \left(\frac{1+\varepsilon}{\varepsilon}\right)^{\frac{s}{m}} \cdot \frac{1}{1+\varepsilon} \cdot \frac{z-1}{\left(\frac{1+\varepsilon}{\varepsilon}\right)^{\frac{1}{m}} - 1}.$$

Then we determine the difference between the contribution to $V_{\min}(d_j)$ by the new sequence and the original geometric sequence. We calculate this difference for the deadlines at position s

$$\begin{aligned}\Delta(s) &= d_j - d_j \cdot \left(\frac{1+\varepsilon}{\varepsilon}\right)^{\frac{s}{m}} \cdot \left(1 - \frac{1}{1+\varepsilon}\right) - \left(d_j - \left(d_j \cdot \left(\frac{1+\varepsilon}{\varepsilon}\right)^{\frac{s-1}{m}} \cdot z - p_i\right)\right) \\ &= d_j \cdot \frac{z - \left(\frac{1+\varepsilon}{\varepsilon}\right)^{\frac{1}{m}}}{\left(\frac{1+\varepsilon}{\varepsilon}\right)^{\frac{1}{m}} - 1} \cdot \left(\left(\frac{1+\varepsilon}{\varepsilon}\right)^{\frac{s-m}{m}} - \left(\frac{1+\varepsilon}{\varepsilon}\right)^{\frac{s-1}{m}}\right)\end{aligned}$$

and for a deadline at position q with $s+1 \leq q \leq m$:

$$\begin{aligned}\Delta(q) &= d_j - d_j \cdot \left(\frac{1+\varepsilon}{\varepsilon}\right)^{\frac{q}{m}} \cdot \left(1 - \frac{1}{1+\varepsilon}\right) - \left(d_j - d_j \cdot \left(\frac{1+\varepsilon}{\varepsilon}\right)^{\frac{q-1}{m}} \cdot z \cdot \left(1 - \frac{1}{1+\varepsilon}\right)\right) \\ &= d_j \cdot \left(\frac{1+\varepsilon}{\varepsilon}\right)^{\frac{q-1}{m}-1} \cdot \left(z - \left(\frac{1+\varepsilon}{\varepsilon}\right)^{\frac{1}{m}}\right)\end{aligned}$$

The total difference over the sequence of deadlines is

$$\begin{aligned}\sum_{q=s+1}^{m} \Delta(q) &= d_j \cdot \left(z - \left(\frac{1+\varepsilon}{\varepsilon}\right)^{\frac{1}{m}}\right) \cdot \frac{\varepsilon}{1+\varepsilon} \cdot \sum_{q=s+1}^{m} \left(\frac{1+\varepsilon}{\varepsilon}\right)^{\frac{q-1}{m}} \\ &= d_j \cdot \frac{z - \left(\frac{1+\varepsilon}{\varepsilon}\right)^{\frac{1}{m}}}{\left(\frac{1+\varepsilon}{\varepsilon}\right)^{\frac{1}{m}} - 1} \cdot \left(1 - \left(\frac{1+\varepsilon}{\varepsilon}\right)^{\frac{s}{m}-1}\right).\end{aligned}$$

$$\sum_{q=s}^{m} \Delta(q) = d_j \cdot \frac{z - \left(\frac{1+\varepsilon}{\varepsilon}\right)^{\frac{1}{m}}}{\left(\frac{1+\varepsilon}{\varepsilon}\right)^{\frac{1}{m}} - 1} \cdot \left(\left(\frac{1+\varepsilon}{\varepsilon}\right)^{\frac{s-m}{m}} - \left(\frac{1+\varepsilon}{\varepsilon}\right)^{\frac{s}{m}} + 1 - \left(\frac{1+\varepsilon}{\varepsilon}\right)^{\frac{s-m}{m}}\right) \geq 0.$$

Due to the restrictions on z, we have $\sum_{q=s}^{m} \Delta_{\min}(q) = 0$ if and only if $s = 0$ holds. Therefore, we obtain the largest contribution to $V_{\min}(d_j)$ with a sequence of geometrically increasing deadlines. For this sequence, we calculate $V_{\min}(d_j)$:

$$V_{\min}(d_j) = d_j \cdot f(m,\varepsilon) + \sum_{i=1}^{m-1} \left(d_j - \frac{\varepsilon}{1+\varepsilon} \cdot \left(\frac{1+\varepsilon}{\varepsilon}\right)^{\frac{i}{m}} \cdot d_j\right) = d_j \cdot m$$

This result contradicts our assumption. ◀

Algorithm 1 Admission_Control

```
1: for each newly submitted job J_j do
2:     tentatively accept J_j
3:     for each accepted job J_i do
4:         if d_i ≥ d_j then
5:             calculate v = V_accept(d_i)
6:             if v > f(m, ε) · d_i then
7:                 reject J_j
```

4 Upper Bound of the Online Algorithm

In this section we present our online algorithm for the $P_m|\varepsilon, \text{online}, \text{pmtn}|\sum p_j(1-U_j)$ problem. First we address the admission control case with all jobs arriving at time 0, i.e., the online property only considers the sequence of jobs but no progression of time. Our Algorithm 1 Admission_Control simply applies the threshold of Lemma 5: If no accepted job (including the newly submitted one) exceeds the threshold then the job is accepted, otherwise it is rejected. Since no job is large, a job J_j does not influence $V_{\text{accept}}(t)$ for $t < d_j$. Due to the first paragraph of the proof of Lemma 5, we only must consider time instances that are deadlines. Having accepted the jobs, a schedule can be generated by applying Lemma 4. Next we prove the competitive factor of this algorithm.

▶ **Theorem 6.** *The $P_m|\varepsilon, online, pmtn|\sum p_j\cdot(1-U_j)$ problem with $r_j = 0$ for all jobs admits a deterministic online admission control algorithm with competitive ratio at most $\frac{m\cdot(1+\varepsilon)}{\sum_{i=0}^{m-1} g(\frac{i}{m})}$.*

Proof. Since any algorithm produces an optimal result if it accepts all jobs we assume that some jobs are rejected. Since the deadline condition Inequality (4) is valid for every submitted job J_j, algorithm Admission_Control only rejects a job J_j if there is a $t \geq d_j$ with

$$\begin{aligned} V_{\text{accept}}(t) &\geq t\cdot f(m,\varepsilon) - p_j \geq t\cdot\left(f(m,\varepsilon) - \frac{1}{1+\varepsilon}\right) \\ &\geq \frac{t}{1+\varepsilon}\cdot\left(\varepsilon\cdot\sum_{h=1}^{m}\left(\frac{1+\varepsilon}{\varepsilon}\right)^{\frac{h}{m}} - 1\right) = \frac{t}{1+\varepsilon}\cdot\sum_{h=0}^{m-1} g(\frac{h}{m}) \end{aligned}$$

Due to Equations (3) and (6), we have $V_{\min}(\tau) \geq V_{\text{accept}}(\tau)$ for all $\tau \geq 0$ while no algorithm can yield more than $V_{\min}(\tau) = \tau\cdot m$. In the following we consider two intervals: $I_1 = [0, t_s)$ with t_s being the largest time with $V_{\min}(t) = t\cdot\frac{1}{1+\varepsilon}\cdot\sum_{h=0}^{m-1} g(\frac{h}{m})$. Interval $I_2 = [t_s, d_\nu)$ only exists for $d_\nu > t_s$. If this case the maximum total processing in I_2 is $V' = V_{\min}(d_\nu) - V_{\min}(t_s)$ otherwise we say $V' = 0$. Note that $t_s > d_j$ holds. Since there is no rejected job with deadline t_s or larger it is not possible to increase V' in an optimal schedule. Then we have

$$c_A \leq \frac{t_s\cdot m + V'}{t_s\cdot\frac{1}{1+\varepsilon}\cdot\sum_{h=0}^{m-1} g(\frac{h}{m}) + V'} < \frac{t_s\cdot m}{t_s\cdot\frac{1}{1+\varepsilon}\cdot\sum_{h=0}^{m-1} g(\frac{h}{m})} = \frac{m}{\frac{1}{1+\varepsilon}\cdot\sum_{h=0}^{m-1} g(\frac{h}{m})}\,. \qquad \blacktriangleleft$$

For the general case with progression of time we use Algorithm 2 Online_Utilization. Lines 9 to 12 of Algorithm Online_Utilization are identical to Algorithm Admission_Control. Our reference time t_{ref} is the latest submission time (Line 2). Since progression of time may produce large jobs we must also examine the acceptance condition for those time instances at which the large part of a job ends due to the first paragraph of the proof of Lemma 5, see

Algorithm 2 Online_Utilization

```
1:  for each newly submitted job J_j do
2:      t_ref = r_j;
3:      U_j = 0;
4:      for each job J_i with U_i = 0 do
5:          if p_i > (d_i − t_ref)/(1 + ε) and (d_i − t_ref)/(1 + ε) + t_ref ≥ d_j then
6:              calculate v = V_accept((d_i − t_ref)/(1 + ε) + t_ref)
7:              if v > f(m, ε) · (d_i − t_ref)/(1 + ε) then
8:                  U_j = 1
9:          if d_i ≥ d_j then
10:             calculate v = V_accept(d_i)
11:             if v > f(m, ε) · (d_i − t_ref) then
12:                 U_j = 1
13:         if d_i > d_j − p_j and d_i < d_j then
14:             calculate v = V_min(d_i)
15:             if v > m · (d_i − t_ref) then
16:                 U_j = 1
```

Lines 5 to 8. We must use the current reference time when determining these time instances (Lines 5 and 6).

Progression of the reference time may lead to large jobs and it may produce a violation of the acceptance condition for some time instances. To prevent such a situation, we introduce the additional *legal test* based on Lemma 4 for all jobs that have passed the acceptance test of Lemma 5, see Lines 13 to 16 in Algorithm 2 Online_Utilization. If a job passes all tests it is accepted.

▶ **Lemma 7.** *There is a legal schedule for all jobs accepted by Algorithm 2 Online_Utilization.*

Proof. The proof directly follows from Lemma 4. ◀

In the next theorem we derive the competitive factor of our algorithm.

▶ **Theorem 1** (restated). *The $P_m|\varepsilon, online, pmtn|\sum p_j \cdot (1 - U_j)$ problem admits a deterministic online algorithm with competitive ratio at most* $\max\{\frac{m\cdot(1+\varepsilon)}{\sum_{i=0}^{m-1} g(\frac{i}{m})}, \frac{4}{3}\}$.

Proof. We use induction on the number of submission times. For a single submission time, the claim holds due to Theorem 6. Therefore, we assume validity of the claim for k different submission times. As in the proof of Theorem 6 let t_s be the largest time instance with $V_{\min}(t_s) = t_s \cdot \frac{1}{1+\varepsilon} \cdot \sum_{h=0}^{m-1} g(\frac{h}{m})$ with respect to any previous reference time (release date of a job). If we use the expression *with respect to a reference time* then we reduce all times by the reference time when calculating $V_{\min}(t)$ and $V_{\text{accept}}(t_s)$ while we keep the original values for the purpose of time comparison. We divide the rest of our proof into four cases.

1. There is no rejection of any job with release date r_{k+1} due to the legal test in Algorithm 2 Online_Utilization and $t_s \geq r_{k+1}$. Since the progression of the reference time to r_{k+1} cannot increase t_s in a non-delay schedule without the acceptance of a new job we can assume that all job parts that are not completed before r_{k+1} have release date r_{k+1} and apply the proof of Theorem 6 to all jobs with release date r_{k+1}.

2. There is no rejection of any job with release date r_{k+1} due to the legal test in Algorithm 2 Online_Utilization and $t_s < r_{k+1}$. We assume that the optimal schedule uses all available resources before t_s. Then we define V' as in the proof of Theorem 6 and split all job parts contributing to V' into a job part that can be executed before r_{k+1} and a job part that must be executed after r_{k+1}. We denote the total processing time of the first type of job parts by $V'(t_s)$. If our scheduling algorithm executes the $V'(t_s)$ completely within interval $[t_s, r_{k+1})$ then we can apply the proof of Theorem 6 to all jobs with release date r_{k+1} and obtain the claim. Therefore, we assume that not the total processing time of $V'(t_s)$ is scheduled within interval $[t_s, r_{k+1})$. Such situation is only possible if a job J_j from $V'(t_s)$ with processing time p_j is started at time $\tau > r_{k+1} - p_j$. In a non-delay schedule such delayed start requires all machines to be busy in interval $[t_s, \tau)$. In order to maximize the total processing time of $V'(t_s)$ that is scheduled after r_{k+1} we assume that we have b long jobs each with the maximum processing time $r_{k+1} - t_s$ and each starting at time τ while all other jobs contributing to $V'(t_s)$ are only executed in interval $[t_s, \tau)$, that is, we have $V'(t_s) = (\tau - t_s) \cdot m + (r_{k+1} - t_s) \cdot b \leq m \cdot (r_{k+1} - t_s)$. We denote the total processing time of jobs with release date r_{k+1} in our schedule including $V' \backslash V'(t_s)$ by $\bar{V}$. Due to Theorem 6, the total processing time in the optimal schedule cannot exceed $c_A \cdot (\bar{V} + b \cdot (\tau - t_s))$. This situation cannot influence the competitive factor if the following condition holds:

$$\begin{aligned} c_A &\geq \frac{c_A \cdot (\bar{V} + b \cdot (\tau - t_s)) + b \cdot (r_{k+1} - t_s) + m \cdot (\tau - t_s)}{\bar{V} + b \cdot (r_{k+1} - t_s) + m \cdot (\tau - t_s)} \\ \Leftrightarrow 1 - \frac{1}{c_A} &\geq \frac{\frac{b}{m} \cdot \frac{\tau - t_s}{r_{k+1} - t_s}}{\frac{b}{m} + \frac{\tau - t_s}{r_{k+1} - t_s}} \end{aligned}$$

The term on the right side has the maximum value for $\frac{b}{m} = \frac{\tau - t_s}{r_{k+1} - t_s} = \frac{1}{2}$ resulting in $c_A \geq \frac{4}{3}$, see also Hussein and Schwiegelshohn [12]. Therefore, the competitive factor is not affected for small ε.

3. A least one job with release date r_{k+1} is rejected due to the legal test in Algorithm 2 Online_Utilization and $t_s \leq r_{k+1}$. This case leads to a contradiction since we have $V_{\text{accept}}(t) \leq V_{\min}(t) \leq t \cdot \frac{1}{1+\varepsilon} \cdot \sum_{h=0}^{m-1} g(\frac{h}{m})$ for all $t \geq r_{k+1}$ with respect to reference time r_{k+1}. Therefore, we can apply Lemma 5 for all jobs with release date r_{k+1} and any accepted job with release date r_{k+1} passes the legal test.

4. A least one job with release date r_{k+1} is rejected due to the legal test in Algorithm 2 Online_Utilization and $t_s \leq r_{k+1}$. Before discussing this case we consider the worst case example in the proof of Lemma 5 for $V_{\min}(t_s) = t_s \cdot \frac{1}{1+\varepsilon} \cdot \sum_{h=0}^{m-1} g(\frac{h}{m})$ and a single release time. Let $t_0 = t_s \cdot \frac{\varepsilon}{1+\varepsilon}$. Then we have for $t \geq t_0$

$$\begin{aligned} V_{\min}(t) &\leq m \cdot \left(\int_{t_0}^{t} \left(1 - \log_{\frac{1+\varepsilon}{\varepsilon}} \left(\frac{y}{t_0} \right) \right) dy + t_0 \right) \\ &\leq m \cdot \left(t - \frac{1}{\ln \frac{1+\varepsilon}{\varepsilon}} \cdot \left(t \cdot \ln \frac{t}{t_0} - t + t_0 \right) \right). \end{aligned} \tag{8}$$

Progression of time without accepting any new jobs can only increase the bound for $V_{\min}(t)$. Since any new accepted job must also pass the acceptance test of Lemma 5, Inequality (8) must always hold for a possibly new t_s. Therefore, any job J_h with release date r_{k+1} and a deadline $d_h \geq t_s$ cannot fail the legal test. ◀

5 Lower Bounds for Deterministic Online Algorithms

▶ **Theorem 2** (restated). *Any deterministic online algorithm for the* $P_m|\varepsilon, online, pmtn|\sum p_j \cdot (1-U_j)$ *problem has a competitive ratio of at least* $\frac{\lfloor m\cdot(1+\varepsilon)\rfloor}{\sum_{i=0}^{m-1} g(\frac{i}{m})} \cdot (1-\delta)$ *for any* $\delta > 0$.

Proof. In our proof the job sequence consists of several series of jobs all with $d_j \geq p_j \cdot (1+\varepsilon)$. In every step the adversary submits identical jobs until we have either accepted the planned number of jobs or until $\lfloor m \cdot (1+\varepsilon)\rfloor$ jobs have been submitted. We will show that the latter case produces a competitive ratio $c_A \geq \frac{\lfloor m\cdot(1+\varepsilon)\rfloor}{\sum_{i=0}^{m-1} g(\frac{i}{m})}$. Therefore, we are forced to accept the desired number of jobs.

Furthermore, we must show that there is a legal schedule for all accepted jobs. To this end, we use Lemma 4 with deadlines $d_i = \frac{1+\varepsilon}{\varepsilon} \cdot g(\frac{i-1}{m}) = g(\frac{i-1}{m}+1)$ for $1 \leq i \leq m$ and $d_{m+1} = (1+\varepsilon)\cdot(\frac{1+\varepsilon}{\varepsilon} - \delta)$ for an arbitrarily small $\delta > 0$. Since there is only a single accepted job for every deadline d_i with $1 < i \leq m$, less than m jobs contribute to the total processing time in $V_{\min}(d_{i+1}) - V_{\min}(d_i)$ for $1 \leq i < m$. Therefore, we must only consider $V_{\min}(d_1)$ to show the existence of a legal schedule.

1. The adversary submits a job with processing time $\sum_{i=0}^{m-1} g(\frac{i}{m}) - \lfloor \sum_{i=0}^{m-1} g(\frac{i}{m})\rfloor < 1$ and deadline $(1+\varepsilon)$. We must accept this job to prevent $c_A \to \infty$.
2. The adversary submits identical jobs with processing time 1 and deadline $1+\varepsilon$ until $\lfloor \sum_{i=0}^{m-1} g(\frac{i}{m})\rfloor + 1$ such jobs have been accepted. If we accept this number of jobs then our total processing time is
$$\sum_{J_j \in \mathcal{J}} (1-U(j))\cdot p_j \quad = \quad \sum_{i=0}^{m-1} g(\frac{i}{m}) + 1 = \sum_{i=1}^{m} g(\frac{i}{m}) = V_{\min}(1+\varepsilon),$$
see Equation (1). There is a legal schedule since we have $V_{\min}(1+\varepsilon) < m\cdot g(1) = m\cdot(1+\varepsilon)$. If we accept at most $\lfloor \sum_{i=0}^{m-1} g(\frac{i}{m})\rfloor$ of these jobs then we have a total processing time $\sum_{J_j\in\mathcal{J}}(1-U(j))\cdot p_j = \sum_{i=0}^{m-1} g(\frac{i}{m})$ and a competitive factor
$$c_A \quad = \quad \frac{\lfloor m\cdot(1+\varepsilon)\rfloor}{\sum_{i=0}^{m-1} g(\frac{i}{m})}.$$
3. The adversary executes $m-1$ similar submission iterations. We assume that at the beginning of iteration k we have
$$\sum_{J_j \in \mathcal{J}} (1-U(j))\cdot p_j \quad = \quad \sum_{i=k}^{m+k-1} g(\frac{i}{m}) \text{ and}$$
$$V_{\min}(1+\varepsilon) \quad = \quad \sum_{i=1}^{m} g(\frac{i}{m}) + (k-1)\cdot(1+\varepsilon) - \sum_{i=1}^{k-1} g(\frac{i}{m}),$$
respectively. Clearly, this assumption holds for $k=1$. The adversary submits jobs with processing time $p = \frac{1}{\varepsilon} g(\frac{k}{m})$ and deadline $d = \frac{1+\varepsilon}{\varepsilon} g(\frac{k}{m}) = g(\frac{k}{m}+1)$ until we accept one job. If we do not accept any of these jobs then the total processing time remains unchanged and we obtain
$$c_A \quad = \quad \frac{\lfloor m\cdot(1+\varepsilon)\rfloor \cdot \frac{1}{\varepsilon} g(\frac{k}{m})}{\sum_{i=k}^{m+k-1} g(\frac{i}{m})} = \frac{\lfloor m\cdot(1+\varepsilon)\rfloor}{\sum_{i=0}^{m-1} g(\frac{i}{m})}.$$
Otherwise we have
$$\sum_{J_j \in \mathcal{J}} (1-U(j))\cdot p_j \quad = \quad \sum_{i=k}^{m+k-1} g(\frac{i}{m}) + \frac{1}{\varepsilon}\cdot g(\frac{k}{m}) = \sum_{i=k+1}^{m+k} g(\frac{i}{m})$$
$$\Leftrightarrow V_{\min}(1+\varepsilon) \quad = \quad \sum_{i=1}^{m} g(\frac{i}{m}) + k\cdot(1+\varepsilon) - \sum_{i=1}^{k} g(\frac{i}{m})$$
due to $1+\varepsilon-(d-p) = 1+\varepsilon-g(\frac{k}{m})$ and Equation (1).

4. After iteration $m-1$ we have

$$\sum_{J_j \in \mathcal{J}} (1-U(j)) \cdot p_j = \sum_{i=m}^{2m-1} g(\frac{i}{m}) \text{ and}$$

$$V_{\min}(1+\varepsilon) = g(1) + (m-1) \cdot (1+\varepsilon) = m \cdot (1+\varepsilon).$$

Finally, the adversary submits $\lfloor m \cdot (1+\varepsilon) \rfloor$ jobs with processing time $p = \frac{1+\varepsilon}{\varepsilon} \cdot (1-\delta)$ and deadline $d_{m+1} = (1+\varepsilon) \cdot \frac{1+\varepsilon}{\varepsilon} \cdot (1-\delta)$ for an arbitrarily small $\delta > 0$. Since any such job must start at the latest at time

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

we cannot accept any such job and obtain the competitive factor

$$c_A = \frac{\lfloor m \cdot (1+\varepsilon) \rfloor \cdot \frac{1+\varepsilon}{\varepsilon} \cdot (1-\delta)}{\sum_{i=m}^{2m-1} g(\frac{i}{m})} = \frac{\lfloor m \cdot (1+\varepsilon) \rfloor}{\sum_{i=0}^{m-1} g(\frac{i}{m})} \cdot (1-\delta). \quad \blacktriangleleft$$

We now turn our attention to the greedy acceptance policy. Here, we accept any job that can be computed prior to its deadline without delaying the currently accepted jobs beyond their respective deadlines.

▶ **Theorem 3** (restated). *The greedy acceptance policy for the $P_m|\varepsilon, online, pmtn|\sum p_j \cdot (1-U_j)$ problem has a competitive ratio of at least $\frac{1+\varepsilon}{\varepsilon} - \delta$ for any $\delta > 0$.*

Proof. We consider an arbitrarily small $\delta > 0$ and use at the beginning a sequence of $1 + \lceil m \cdot (1+\varepsilon) \rceil$ jobs with the following processing times:

$$p_1 = \varepsilon \cdot m \cdot \delta$$

$$p_2 = \begin{cases} m \cdot (1+\varepsilon) - \lfloor m \cdot (1+\varepsilon) \rfloor - \varepsilon \cdot m \cdot \delta & \text{for } m \cdot (1+\varepsilon) \neq \lfloor m \cdot (1+\varepsilon) \rfloor \\ 1 - \varepsilon \cdot m \cdot \delta & \text{for } m \cdot (1+\varepsilon) = \lfloor m \cdot (1+\varepsilon) \rfloor \end{cases}$$

$$p_3 = p_4 = \ldots = p_{1+\lceil m \cdot (1+\varepsilon) \rceil} = 1$$

All jobs $p_1, \ldots p_{1+\lceil m \cdot (1+\varepsilon) \rceil}$ have deadline $1+\varepsilon$ and must be accepted according to our policy since we have

$$V_{\min}(1+\varepsilon) = \sum_{i=1}^{1+\lceil m \cdot (1+\varepsilon) \rceil} p_i = m \cdot (1+\varepsilon). \tag{9}$$

Then the adversary submits m identical jobs with processing times $p = \frac{1+\varepsilon}{\varepsilon} - \delta$ and deadline $d = (1+\varepsilon) \cdot p = \frac{(1+\varepsilon)^2}{\varepsilon} - (1+\varepsilon) \cdot \delta$. Since the acceptance of any one of these jobs leads to

$$V_{\min}(1+\varepsilon) = m \cdot (1+\varepsilon) + 1 + \varepsilon - (d-p) = m \cdot (1+\varepsilon) + \varepsilon \cdot \delta > m \cdot (1+\varepsilon) \tag{10}$$

we must reject everyone of these jobs.

However, the rejection of job J_1 allows acceptance of all other jobs since we have $V_{\min}(1+\varepsilon) = m \cdot (1+\varepsilon)$ and $V_{\min}(\frac{(1+\varepsilon)^2}{\varepsilon} - (1+\varepsilon) \cdot \delta) = m \cdot \left(\frac{(1+\varepsilon)^2}{\varepsilon} - (1+\varepsilon) \cdot \delta \right)$ due to Equations (9) and (10).

Therefore, the greedy acceptance algorithm has a competitive factor of at least

$$c_{A_{\text{greedy}}} = \frac{m \cdot \left(\frac{(1+\varepsilon)^2}{\varepsilon} - (1+\varepsilon) \cdot \delta \right)}{m \cdot (1+\varepsilon)} = \frac{1+\varepsilon}{\varepsilon} - \delta. \quad \blacktriangleleft$$

Since the lower bound of the competitive factor for the common preemption model is identical to the competitive factor for preemption without migration, see Lemma 5 and DasGupta and Palis [7], we can state that a greedy acceptance policy cannot exploit the benefits of migration.

References

1. S. Albers and M. Hellwig. On the value of job migration in online makespan minimization. In *Proc. of ESA*, pages 84–95, 2012.
2. J. H. Anderson, V. Bud, and U. C. Devi. An EDF-based restricted-migration scheduling algorithm for multiprocessor soft real-time systems. *Real-Time Systems*, 38(2):85–131, 2008.
3. S. K. Baruah, G. Koren, D. Mao, B. Mishra, A. Raghunathan, L. E. Rosier, D. Shasha, and F. Wang. On the competitiveness of on-line real-time task scheduling. *Real-Time Systems*, 4(2):125–144, 1992.
4. S.K. Baruah and J.R. Haritsa. Scheduling for overload in real-time systems. *IEEE Trans. Computers*, 46(9):1034–1039, 1997.
5. P. Brucker and S. Knust. Complexity results for scheduling problems. `http://www2.informatik.uni-osnabrueck.de/knust/class/`, 2009. [Online; accessed 11-April-2016].
6. B. Chen, A. van Vliet, and G. J. Woeginger. An optimal algorithm for preemptive on-line scheduling. *Oper. Res. Lett.*, 18(3):127–131, 1995.
7. B. DasGupta and M.A. Palis. Online real-time preemptive scheduling of jobs with deadlines on multiple machines. *Journal of Scheduling*, 4(6):297–312, 2001.
8. R. I. Davis and A. Burns. A survey of hard real-time scheduling for multiprocessor systems. *ACM Comput. Surv.*, 43(4):35, 2011.
9. L. Epstein and A. Levin. Robust algorithms for preemptive scheduling. *Algorithmica*, 69(1):26–57, 2014.
10. S.A. Goldman, J. Parwatikar, and S. Suri. Online scheduling with hard deadlines. *Journal of Algorithms*, 34(2):370–389, 2000.
11. M.H. Goldwasser. Patience is a virtue: the effect of slack on competitiveness for admission control. In *Proc. of SODA*, pages 396–405, 1999.
12. M.E. Hussein and U. Schwiegelshohn. Utilization of nonclairvoyant online schedules. *Theor. Comput. Sci.*, 362(1-3):238–247, 2006.
13. B. Kalyanasundaram and K. Pruhs. Eliminating migration in multi-processor scheduling. *J. Algorithms*, 38(1):2–24, 2001.
14. B. Kalyanasundaram and K. Pruhs. Maximizing job completions online. *J. Algorithms*, 49(1):63–85, 2003.
15. T. Kawaguchi and S. Kyan. Worst case bound of an LRF schedule for the mean weighted flow-time problem. *SIAM Journal on Computing*, 15(4):1119–1129, 1986.
16. J.H. Kim and K.Y. Chwa. On-line deadline scheduling on multiple resources. In *Proc. of COCOON*, pages 443–452, 2001.
17. G. Koren and D. Shasha. MOCA: A multiprocessor on-line competitive algorithm for real-time system scheduling. *Theor. Comput. Sci.*, 128(1&2):75–97, 1994.
18. E. L. Lawler. A dynamic programming algorithm for preemptive scheduling of a single machine to minimize the number of late jobs. *Annals of Operations Research*, 26(1):125–133, 1990.
19. J. Lee. Online deadline scheduling: multiple machines and randomization. In *Proc. of SPAA*, pages 19–23, 2003.
20. R.J. Lipton and A. Tomkins. Online interval scheduling. In *Proc. of SODA*, pages 302–311, 1994.
21. M.L. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Springer Science+Business Media, forth edition, 2010.
22. P. Sanders, N. Sivadasan, and M. Skutella. Online scheduling with bounded migration. *Math. Oper. Res.*, 34(2):481–498, 2009.
23. U. Schwiegelshohn. An alternative proof of the Kawaguchi-Kyan bound for the Largest-Ratio-First rule. *Oper. Res. Lett.*, 39(4):255–259, 2011. `doi:10.1016/j.orl.2011.06.007`.

24 M. Skutella and J. Verschae. A robust PTAS for machine covering and packing. In *Proc. of ESA*, pages 36–47, 2010. doi:10.1007/978-3-642-15775-2_4.

25 B. Sotomayor, R. S. Montero, I. M. Llorente, and I. T. Foster. Virtual infrastructure management in private and hybrid clouds. *IEEE Internet Computing*, 13(5):14–22, 2009.

26 P. Valente and G. Lipari. An upper bound to the lateness of soft real-time tasks scheduled by EDF on multiprocessors. In *Proc. of RTSS*, pages 311–320, 2005.

27 G. J. Woeginger. On-line scheduling of jobs with fixed start and end times. *Theor. Comput. Sci.*, 130(1):5–16, 1994.