J Comb Optim (2006) 12:363–386
DOI 10.1007/s10878-006-9005-9

Separating online scheduling algorithms with the relative worst order ratio

Leah Epstein · Lene M. Favrholdt · Jens S. Kohrt

Published online: 20 September 2006

Abstract The relative worst order ratio is a measure for the quality of online algorithms. Unlike the competitive ratio, it compares algorithms directly without involving an optimal offline algorithm. The measure has been successfully applied to problems like paging and bin packing. In this paper, we apply it to machine scheduling. We show that for preemptive scheduling, the measure separates multiple pairs of algorithms which have the same competitive ratios; with the relative worst order ratio, the algorithm which is "intuitively better" is also provably better. Moreover, we show one such example for non-preemptive scheduling.

Keywords Online algorithms · Relative worst order ratio · Scheduling

1 Introduction

The relative worst order ratio is a relatively new quality measure for online algorithms, inspired by the Max/Max ratio (Ben-David and Borodin, 1994) and the random order ratio (Kenyon, 1996) and defined in Boyar and Favrholdt (2003). Since it compares two algorithms directly, it sometimes gives more detailed information than the competitive ratio. So far, the results on the relative worst order ratio have been either consistent with competitive analysis or closer to empirical results and/or intuition than results on the competitive ratio.

L. Epstein (✉)
Department of Mathematics, University of Haifa, Israel
e-mail: lea@math.haifa.ac.il

L. M. Favrholdt · J. S. Kohrt
Department of Mathematics and Computer Science, University of Southern Denmark, Odense, Denmark

L. M. Favrholdt
e-mail: lenem@imada.sdu.dk

J. S. Kohrt
e-mail: svalle@imada.sdu.dk

In this paper we analyze several scheduling problems where the competitive ratio does not give the right separation of algorithms, in the hope that the relative worst order ratio would do better. In most of the cases considered, the relative worst order ratio prefers the intuitively best algorithm whereas the competitive ratio does not distinguish the algorithms. Analyzing the relative worst order ratio is often more straightforward than competitive analysis. Unlike the optimal offline algorithm, which is generally not known, we know exactly what the two compared algorithms do on a given input. In many cases, this can simplify the analysis.

We first define the two quality measures.

The quality measures. For any algorithm A and any sequence σ of jobs, let $\mathrm{A}(\sigma)$ be the makespan obtained by A when scheduling σ. Similarly, let $\mathrm{OPT}(\sigma)$ be the makespan obtained by an optimal offline algorithm.

The general definition of the *competitive ratio* of an algorithm A is

$$\mathrm{CR_A} = \inf\{c \mid \exists b\colon \forall \sigma\colon \mathrm{A}(\sigma) \leq c \cdot \mathrm{OPT}(\sigma) + b\}.$$

However, scheduling problems are typically scalable. This means that all job lengths of an adversarial instance can be scaled by any factor, and thus the additive constant b has no effect. Hence, for the problems considered in this paper, the definition reduces to

$$\mathrm{CR_A} = \sup_{\sigma} \frac{\mathrm{A}(\sigma)}{\mathrm{OPT}(\sigma)}.$$

For the problems considered in this paper, the *relative worst order ratio* is defined in the following way (a general, and thus slightly more involved, definition can be found in Boyar et al. (2005)). For any algorithm A and any input sequence σ, let $\mathrm{A_W}(\sigma)$ be the makespan of A on its worst permutation of σ, i.e., $\mathrm{A_W}(\sigma) = \max_p \mathrm{A}(p(\sigma))$, where p is a permutation on $|\sigma|$ elements. If $\mathrm{A_W}(\sigma) \geq \mathrm{B_W}(\sigma)$ for every sequence σ, we say that the two algorithms are *comparable*, and the *relative worst order ratio of* A *to* B is

$$\mathrm{WR_{A,B}} = \sup_{\sigma} \frac{\mathrm{A_W}(\sigma)}{\mathrm{B_W}(\sigma)}.$$

For some pairs of algorithms, there are sequences σ such that $\mathrm{A_W}(\sigma) > \mathrm{B_W}(\sigma)$ and other sequences σ' such that $\mathrm{A_W}(\sigma') < \mathrm{B_W}(\sigma')$. In this case we say that the two algorithms are *incomparable*. For comparable algorithms, the measure is transitive. Specifically, it is shown in Boyar and Favrholdt (2003) that given three algorithms A, B, C such that $\mathrm{WR}_{C,B} \geq 1$ and $\mathrm{WR}_{B,A} \geq 1$, then $\mathrm{WR}_{C,A} \geq 1$ and moreover $\min\{\mathrm{WR}_{C,B}, \mathrm{WR}_{B,A}\} \leq \mathrm{WR}_{C,A} \leq \mathrm{WR}_{C,B} \cdot \mathrm{WR}_{B,A}$.

The relative worst order ratio has previously been applied to bin packing (Boyar and Favrholdt, 2003), paging (Boyar et al., 2005), seat reservation (Boyar and Medvedev, 2004), and bin coloring (Kohrt, 2004). In this paper, the measure is applied to scheduling.

The scheduling problems. In the basic scheduling problem, we are given m machines and a sequence of jobs, each characterized by the time it takes to execute it on a unit speed machine. For problems where the machines have different speeds, a job of size p requires time $\frac{p}{s}$ when run on a machine of speed s. The *load* of a machine is the total size of jobs or parts of jobs scheduled on this machine. The goal is to minimize the *makespan*, i.e., the time when all jobs are completed. The jobs arrive one by one. Each job must be scheduled at arrival, and this schedule cannot be changed afterwards. In the *non-preemptive* case, a job has to run without interruption on a single machine. In the *preemptive* case, the algorithms are allowed

to preempt the job and run parts of it on different machines, as long as two parts of a job are never run at the same time. For preemptive algorithms it may make sense to use idle time. However, the algorithms stated and defined in this paper do not use idle time.

We study four scheduling problems, preemptive and non-preemptive scheduling on identical machines and on two uniformly related machines.

Results. We first consider the preemptive problems. For *identical* machines, we define a class of algorithms. This class generalizes two previously known algorithms, those of Seiden (2001) and of Chen et al. (1995). For any pair of algorithms in this class, one algorithm A is better than the other algorithm B in the sense that A is never worse than B and on some sequences it is better. In contrast to the competitive ratio which is the same for all these algorithms, the relative worst order ratio shows this separation. The previously known algorithms are the two extremes of this class, being the best and the worst algorithms in the class. For *two uniformly related* machines we again consider two previously known algorithms, of Wen and Du (1998) and Epstein et al. (2001). We generalize the first of them into a class of algorithms, all having the optimal competitive ratio. All the new algorithms in the class turn out to be better than the original one. We compare all these algorithms to the second one, and find that this algorithm is strictly better than all the algorithms in the class. Again, we show a clear separation between any two algorithms in the presented class.

For non-preemptive scheduling on two related machines, we again show a clear separation between two algorithms having the same competitive ratio. For non-preemptive scheduling on identical machines, the three algorithms considered are shown to be incomparable.

2 Preemptive scheduling to minimize makespan

We use the following notation. For any sequence σ of n jobs with sizes $p_1, p_2, \ldots, p_n$, we let P and $p_{\max}$ denote the total and maximum size of the jobs in σ, respectively. Similarly, P^t and $p^t_{\max}$ denote the total and maximum size of the first t jobs $J_1, J_2, \ldots, J_t$ in σ. For any machine m_i and any job J_t, L^t_i denotes the load of machine m_i just after scheduling $J_1, J_2, \ldots, J_t$. For any algorithm A, A(σ) denotes the makespan of A's schedule for σ. In particular, OPT(σ) denotes the makespan of an optimal offline algorithm OPT.

2.1 Identical machines

For the scheduling problem studied in this section, we have $m \geq 2$ identical machines available, and preemption is allowed. We study two known algorithms and a generalization of the two.

Two algorithms with optimal competitive ratio. Two online algorithms with optimal competitive ratio have been suggested for this problem, PREEMPTIVE (PRE) by Chen (1995) and MODIFIED PREEMPTIVE (MPRE) by Seiden (2001). In short, they both keep track of a maximal allowable makespan M^t, where M^t is defined differently for the two algorithms. The algorithm MPRE is defined so that it uses at most one preemption per job. The other algorithm PRE is defined to use up to $m - 1$ preemptions for each job. It is explained in Chen et al. (1995), however, that it is not difficult to adapt it so that it also uses at most one preemption per job. Moreover, with the same definition of M^t, the two algorithms would maintain the same set of loads.

MPRE schedules each job J_t in the following way. Let m be the currently most loaded machine just before J_t is scheduled. The job is scheduled completely on m, if this gives a

makespan of at most M^t. Otherwise, as much as possible of the job is scheduled on the least loaded machine m' among those on which the job cannot finish earlier than M^t, and the rest of it is scheduled on the most loaded machine among those that have load less than m', giving a makespan of exactly M^t.

Let $m_1, \ldots, m_m$ be the sequence of machines sorted by non-decreasing load. PRE assigns the $(t+1)^{st}$ job J_{t+1} as follows. First, a new maximal allowable makespan M^{t+1} is computed. Then, on each machine m_j, the time interval I_j is reserved for job J_{t+1}, where

$$I_m = \left[L_m^t\ ,\ M^{t+1}\right] \quad \text{and} \quad I_j = \left[L_j^t\ , L_{j+1}^t\right],\ \text{for } 1 \le j \le m-1.$$

Those intervals are disjoint. The total processing time that can be assigned on all intervals is

$$\sum_{j=1}^{m-1} \left(L_{j+1}^t - L_j^t\right) + M^{t+1} - L_m^t = M^{t+1} - L_1^t.$$

To assign J_{t+1}, go from I_m to I_1, putting a part of the job, as large as possible in each interval, until all the job is assigned. After the assignment there will be some fully occupied intervals $I_{z+1}, \ldots, I_m$, some empty intervals $I_1, \ldots, I_{z-1}$ and a partially or fully occupied interval I_z.

Chen et al. (1995) prove that using this strategy always results in a feasible schedule, if

$$M_{\text{PRE}}^t = \max\left\{\beta\frac{P^t}{m}\ ,\ \beta\ p_{\max}^t\right\},\ \text{ where }\ \beta = \frac{\theta^m}{\theta^m - 1}\ ,\ \ \theta = \frac{m}{m-1}$$

is used as M^t. Since the optimal offline makespan is $\max\{\frac{P}{m}, p_{\max}\}$ (McNaughton, 1959), the competitive ratio obtained is β. Even if randomization is allowed, this is the optimal competitive ratio. As m approaches infinity, β approaches $\frac{e}{e-1} \approx 1.58$ from below. Note that $\beta/m < 1$, for $m \ge 2$.

Seiden (2001) proves that using

$$M_{\text{MPRE}}^t = \max\left\{\beta\frac{P^t}{m}\ ,\ \frac{m-\beta}{m-1}p_{\max}^t + \frac{\beta-1}{m-1}P^t\right\}$$

also gives a feasible schedule for any job sequence. For any input sequence, the makespan of MPRE is never more, and sometimes less, than the makespan of PRE, since

$$\text{for }\ p_{\max} > \frac{P}{m},\ \ \frac{m-\beta}{m-1}p_{\max} + \frac{\beta-1}{m-1}P < \beta\ p_{\max}.$$

In contrast to the competitive ratio, the relative worst order ratio reflects the fact that MPRE is never worse than PRE and sometimes better (Corollary 1).

A generalized algorithm. We define a generalized algorithm with a parameter b, $0 \le b \le \frac{\beta-1}{m-1}$, ADAPTED PREEMPTIVE$_b$ (APRE$_b$), and use

$$M_b^t = \max\left\{\beta\frac{P^t}{m}\ ,\ (\beta - mb)\ p_{\max}^t + bP^t\right\}$$

as M^t. In the analysis of the algorithm, we assume that the algorithm, like PRE, may use several preemptions per job (but, of course, the results are also valid for the algorithm using at most one preemption per job, like MPRE). Note that, for $b = 0$, $M_b^t = M_{\text{PRE}}^t$, whereas using $b = \frac{\beta-1}{m-1}$ leads to $M_b^t = M_{\text{MPRE}}^t$. Also note that $\beta - mb$ is always positive.

We prove that APRE$_b$ is well-defined and has competitive ratio β (Theorem 1). The relative worst order ratio shows that a larger value of b implies a better algorithm (Theorem 2) even though all these algorithms have the same competitive ratio.

The algorithms maintain the following three invariants. These invariants are the same as the invariants defined in Seiden (2001), except the change of M into M_b^t.

1. At any time t, $L_1^t \le L_2^t \le \cdots \le L_m^t$.
2. At any time t, $L_m^t \le M_b^t$.
3. At any time t, for every $1 \le k \le m$, $\sum_{i=1}^{k} L_i^t \le \frac{\theta^k - 1}{\theta^m - 1} \cdot P^t$

The first two invariants follow from the definition of the algorithms. The third is proved in Lemma 3. First, we use the invariants to show that it is always possible to partition a job among its designated intervals (Lemma 2). Lemma 2 uses the following lemma, showing that which term is maximum in the definition of M_b^t depends only on P^t, $p_{\max}^t$, and m; not on b.

Lemma 1. *After t jobs,* APRE$_b$ *has $M_b^t = \beta \frac{P^t}{m}$ if and only if $\frac{P^t}{m} \ge p_{max}^t$.*

Proof: This follows from

$$\beta \frac{P^t}{m} \ \ge\ (\beta - mb) p_{\max}^t + b P^t \quad \Leftrightarrow \quad \left(\frac{\beta}{m} - b\right) P^t \ \ge\ (\beta - mb) p_{\max}^t \quad \Leftrightarrow \quad \frac{P^t}{m} \ \ge\ p_{\max}^t. \qquad \square$$

Lemma 2. *If the invariants are fulfilled at step t, then the reserved intervals are sufficient to assign J_{t+1}.*

Proof: We consider two cases and show that the assignment is successful in both cases.

– *Case 1:* $p_{t+1} > \frac{P^{t+1}}{m}$. The total size of the reserved intervals is

$$\begin{aligned} M_b^{t+1} - L_1^t \ &\ge\ (\beta - mb)\, p_{\max}^{t+1} + b P^{t+1} - \frac{\theta - 1}{\theta^m - 1} P^t, \\ &\qquad \text{by Lemma 1 } M_b \text{ and the third invariant} \\ &\ge\ (\beta - mb + b) p_{t+1} + \left(b - \frac{\theta - 1}{\theta^m - 1}\right) P^t, \\ &\qquad \text{since } p_{\max}^{t+1} \ge p_{t+1} \text{ and } P^{t+1} = p_{t+1} + P^t. \end{aligned}$$

Note that $b - \frac{\theta-1}{\theta^m-1} = b - \frac{\beta-1}{m-1}$:

$$\begin{aligned} \frac{\beta - 1}{m - 1} \ &=\ \frac{\theta - 1}{\theta^m - 1} \\ \Leftrightarrow \quad \frac{\theta^m}{\theta^m - 1} - 1 \ &=\ \frac{\theta - 1}{\theta^m - 1}(m - 1) \\ \Leftrightarrow \quad \theta^m - (\theta^m - 1) \ &=\ (\theta - 1)(m - 1) \\ \Leftrightarrow \quad 1 = \left(\frac{m}{m - 1} - 1\right)(m - 1) &= m - (m - 1) \end{aligned}$$

Using $P^t < (m-1)p_{t+1}$ (which follows from $p_{t+1} > \frac{P^{t+1}}{m}$ and $P^{t+1} = p_{t+1} + P^t$), we get

$$M_b^{t+1} - L_1^t \;\geq\; (\beta - mb + b)p_{t+1} + \left(b - \frac{\beta-1}{m-1}\right)(m-1)p_{t+1} = p_{t+1}\,.$$

Therefore p_{t+1} can be assigned into the intervals.

– *Case 2:* $p_{t+1} \leq \frac{P^{t+1}}{m}$. The total size of the reserved intervals is

$$M_b^{t+1} - L_1^t \geq \frac{\beta}{m}P^{t+1} - \frac{\beta-1}{m-1}P^t = \frac{\beta}{m}\left(P^t + p_{t+1}\right) - \frac{\beta-1}{m-1}P^t \geq \frac{\beta}{m}p_{t+1} + \frac{m-\beta}{m(m-1)}P^t.$$

Using $P^t \geq (m-1)p_{t+1}$ we get, $M_b^{t+1} - L_1^t \geq p_{t+1}$. Therefore p_{t+1} can be assigned into the intervals in this case as well.

□

To complete the proof that the algorithm is well-defined, we need to show that all invariants are kept after an assignment of a new job. For the first two invariants, this is clear from the definition of the algorithm. Since all loads are initially zero, it suffices to prove the following lemma for the last invariant.

Lemma 3. *If the third invariant is fulfilled after step t, then it is also satisfied after step $t+1$*

Proof: According to the definition of the algorithm, there exists a machine m_z such that for $i < z,\ L_i^{t+1} = L_i^t\,,$ for $z < i \leq m,\ L_i^{t+1} = L_{i+1}^t$, and $L_z^t < L_z^{t+1} \leq L_{z+1}^t$ (for convenience let $L_{m+1}^t = M_b^t$).

– For $k < z$,

$$\sum_{i=1}^{k} L_i^{t+1} = \sum_{i=1}^{k} L_i^t \;\leq\; \frac{\theta^k - 1}{\theta^m - 1}\,P^t \;\leq\; \frac{\theta^k - 1}{\theta^m - 1}\,P^{t+1}\,.$$

– For $k \geq z$, it is sufficient to show the inequality

$$P^{t+1} - \sum_{i=1}^{k} L_i^{t+1} = \sum_{i=k+1}^{m} L_i^{t+1} \;\geq\; \frac{\theta^m - \theta^k}{\theta^m - 1}\,P^{t+1} = P^{t+1} - \frac{\theta^k - 1}{\theta^m - 1}\,P^{t+1}\,.$$

Since $k+1 > z,\ L_i^{t+1} = L_{i+1}^t$, and the left hand side is equal to

$$\begin{aligned}\sum_{i=k+1}^{m} L_i^{t+1} \;&=\; \left(\sum_{i=k+2}^{m} L_i^t\right) + M_b^{t+1} = \left(P^t - \sum_{i=1}^{k+1} L_i^t\right) + M_b^{t+1} \\ &\geq\; \frac{\theta^m - \theta^{k+1}}{\theta^m - 1}\,P^t + M_b^{t+1} \;\geq\; \frac{\theta^m - \theta^{k+1}}{\theta^m - 1}\left(P^{t+1} - p_{\max}^{t+1}\right) + M_b^{t+1}\end{aligned}$$

Hence, it suffices to show

$$M_b^{t+1} \geq \frac{\theta^m - \theta^{k+1}}{\theta^m - 1}\,p_{\max}^{t+1} + \frac{\theta^{k+1} - \theta^k}{\theta^m - 1}\,P^{t+1}.$$

This is proved using that by definition,

$$M_b^{t+1} \geq \frac{\beta P^{t+1}}{m} \quad \text{and} \quad M_b^{t+1} \geq (\beta - mb)p_{\max}^{t+1} + bP^{t+1}.$$

Let $\alpha = \frac{\theta^m - \theta^{k+1}}{(\theta^m - 1)(\beta - mb)}$ and note that $0 \leq \alpha \leq 1$. Multiplying the second inequality is by α and the first by $1 - \alpha$ and adding the two resulting inequalities, we arrive at

$$\begin{aligned}
M_b^{t+1} &\geq \frac{\theta^m - \theta^{k+1}}{\theta^m - 1} p_{\max}^{t+1} + \frac{\theta^m - \theta^{k+1}}{(\theta^m - 1)(\beta - mb)} bP^{t+1} + \left(1 - \frac{\theta^m - \theta^{k+1}}{(\theta^m - 1)(\beta - mb)}\right) \frac{\beta P^{t+1}}{m} \\
&= \frac{\theta^m - \theta^{k+1}}{\theta^m - 1} p_{\max}^{t+1} + \frac{\theta^m - \theta^{k+1}}{(\theta^m - 1)(\beta - mb)} \left(b - \frac{\beta}{m}\right) P^{t+1} + \frac{\beta P^{t+1}}{m} \\
&= \frac{\theta^m - \theta^{k+1}}{\theta^m - 1} p_{\max}^{t+1} - \frac{\theta^m - \theta^{k+1}}{(\theta^m - 1)} \frac{P^{t+1}}{m} + \frac{\beta P^{t+1}}{m} \\
&= \frac{\theta^m - \theta^{k+1}}{\theta^m - 1} p_{\max}^{t+1} - \frac{\theta^m - \theta^{k+1} + \beta(\theta^m - 1)}{(\theta^m - 1)} \frac{P^{t+1}}{m} \\
&= \frac{\theta^m - \theta^{k+1}}{\theta^m - 1} p_{\max}^{t+1} + \frac{\theta^{k+1}}{(\theta^m - 1)} \frac{P^{t+1}}{m}
\end{aligned}$$

Thus, we just need to prove that

$$\frac{\theta^{k+1}}{(\theta^m - 1)} \frac{P^{t+1}}{m} \geq \frac{\theta^{k+1} - \theta^k}{\theta^m - 1} P^{t+1},$$

which is equivalent to

$$\frac{\theta^{k+1}}{m} \geq \theta^{k+1} - \theta^k.$$

Now,

$$\frac{\theta^{k+1}}{m} \geq \theta^{k+1} - \theta^k \;\Leftrightarrow\; \frac{1}{m}\left(\frac{m}{m-1}\right)^{k+1} \geq \left(\frac{m}{m-1}\right)^{k+1} - \left(\frac{m}{m-1}\right)^{k} \;\Leftrightarrow\; 1 \geq 1.$$

This completes the proof of the inequality

$$M_b^{t+1} \geq \frac{\theta^m - \theta^{k+1}}{\theta^m - 1} p_{\max}^{t+1} + \frac{\theta^{k+1} - \theta^k}{\theta^m - 1} P^{t+1},$$

concluding the case $k \geq z$. □

We are now ready to prove Theorem 1.

Theorem 1. *For every* $0 \leq b \leq \frac{\beta - 1}{m - 1}$, APRE$_b$ *is well-defined and has competitive ratio* β.

Proof: By Lemmas 2 and 3 and the discussion before Lemma 3, the algorithm is well-defined.

The competitive ratio is not better than β, since this is the optimal competitive ratio. For the upper bound on the competitive ratio, consider any input sequence σ. If $M_b(\sigma) = \beta\frac{P}{m}$, then clearly the competitive ratio is at most β, since OPT $= \max\{\frac{P}{m}, p_{\max}\}$. Otherwise, by Lemma 1, $P < m\ p_{\max}$ and thus, $M_b(\sigma) < \beta p_{\max}$. □

We can now find the relative worst order ratios of pairs of algorithms with optimal competitive ratio. For this purpose, we prove the following lemma.

Lemma 4. *Let σ be an input sequence with n jobs and let $0 \le b \le \frac{\beta-1}{m-1}$. A permutation, σ_w, of σ where the jobs appear in order of non-increasing sizes is a worst order for* APRE_b*, and*

$$\text{APRE}_b(\sigma_w) = \min\{P,\ M_b(\sigma)\}.$$

Proof: Note that $L_m^{t+1} = \min\{L_m^t + p^{t+1},\ M_b^{t+1}\}$. Since $L_m^t + p^{t+1}$ as well as M_b^{t+1} are maximized when P^{t+1} is maximized, we can prove by induction that the largest makespan after i jobs is achieved if the first i jobs are the largest ones. Therefore, no order can be worse than a non-increasing order. Thus, it is enough to show that a non-increasing order gives $\text{APRE}_b(\sigma) = L_m^n = \min\{P, M_b(\sigma)\}$. Let $P = p_1, \ldots, p_n$ be the sorted list of job sizes.

We prove that if $L_m^t = M_b^t$ then $L_m^{t+1} = M_b^{t+1}$. Assume that $L_m^t = M_b^t$. Then, the interval reserved for job J_{t+1} on the most loaded machine is $M_b^{t+1} - M_b^t$.

– If $M_b^{t+1} = \beta\frac{P^{t+1}}{m}$, then

$$M_b^{t+1} - M_b^t = \beta\frac{P^{t+1}}{m} - M_b^t \ \le\ \beta\frac{P^{t+1}}{m} - \beta\frac{P^t}{m} = \frac{\beta}{m}p_{t+1} < p_{t+1}, \quad \text{since } \frac{\beta}{m} < 1.$$

Thus, the interval is filled completely, giving a makespan of $M_b^{t+1} = \beta\frac{P^{t+1}}{m}$.

– Otherwise, $M_b^{t+1} = (\beta - mb)\, p_{\max}^{t+1} + bP^{t+1}$, and since $p_{\max}^t = p_{\max}^{t+1} = p_1$,

$$M_b^{t+1} - M_b^{t+1} \le bP^{t+1} - bP^t = bp_{t+1} < p_{t+1}, \quad \text{since } b < 1.$$

Thus again, the interval is filled completely, giving a makespan of $M_b^{t+1} = (\beta - mb)\, p_{\max}^{t+1} + bP^{t+1}$.

We are now ready to prove the lemma. As long as jobs are assigned so that the designated interval on the most loaded machine is not filled, there are no jobs assigned to other machines, and the makespan is P^t at time t. Once this interval is filled completely, we showed that it will be filled in the next steps as well. This proves the claim. □

Lemma 5. *For every pair of values $0 \le b_1 < b_2 \le \frac{\beta-1}{m-1}$,* APRE_{b_1} *and* APRE_{b_2} *are comparable, and* APRE_{b_2} *is never worse than* APRE_{b_1}.

Proof: Consider any input sequence σ. By Lemma 4, it is sufficient to prove that $M_{b_2}(\sigma) \le M_{b_1}(\sigma)$, and by Lemma 1, it is sufficient to consider the case when $P < mp_{\max}$. The claim then immediately follows by the definition of M_b, since $M_{b_2}(\sigma) - M_{b_1}(\sigma) = (P - mp_{\max})(b_2 - b_1) < 0$. □

Lemma 6. *For any pair of values $0 \le b_1 < b_2 \le \frac{\beta-1}{m-1}$, $WR_{\text{APRE}_{b_1},\text{APRE}_{b_2}} \ge \dfrac{\beta - mb_1}{\beta(1 + b_2 - b_1) - mb_2}$.*

Proof: By Lemma 5, it is sufficient to find an input sequence giving the stated ratio. Let $\lambda_1 = 1 - b_1$, $\lambda_2 = \beta - 1 - b_1(m-1)$. Note that $\lambda_1 > \lambda_2$, since $2 - b_1(2-m) > \beta$ as $\beta < 2$. Moreover, since $b_1 < \frac{\beta-1}{m-1}$, we have $\lambda_2 > 0$.

Consider the input sequence $\sigma = \langle \lambda_1, \lambda_2 \rangle$. By Lemma 4, it is sufficient to consider this ordering of the two jobs. We have $P = \beta - mb_1$, and $p_{\max} = \lambda_1 = 1 - b_1$. Note that, for any b, we get the same result with APRE_b, if the last job of size λ_2 is split into smaller jobs of total size λ_2.

We have $\frac{P}{p_{\max}} = \frac{\beta - mb_1}{1-b_1} < m$ (since $\beta < m$), and thus $\frac{P}{m} < p_{\max}$. Hence, by Lemma 1, we must have $M_{b_i}(\sigma) = (\beta - mb_i)\, p_{\max} + b_i P$. Now, by Lemma 4, $\text{APRE}_{b_i}(\sigma) = M_{b_i}(\sigma)$ if any only if $(\beta - mb_i)\, p_{\max} + b_i P \le P$. This is equivalent to $\frac{P}{p_{\max}} \ge \frac{\beta - mb_i}{1-b_i}$, which holds for b_1 with equality by the definitions of λ_1 and λ_2. Moreover, $\frac{\beta - mb_i}{1-b_i}$ is a monotonically decreasing function of b_i, and hence $\frac{\beta - mb_2}{1-b_2} \le \frac{\beta - mb_1}{1-b_1}$.

Thus,

$$\frac{\text{APRE}_{b_1}(\sigma)}{\text{APRE}_{b_2}(\sigma)} = \frac{(\beta - mb_1)(\lambda_1) + b_1(\lambda_1 + \lambda_2)}{(\beta - mb_2)(\lambda_1) + b_2(\lambda_1 + \lambda_2)} = \frac{(\beta - mb_1)(1-b_1) + b_1(\beta - mb_1)}{(\beta - mb_2)(1-b_1) + b_2(\beta - mb_1)}$$
$$= \frac{\beta - mb_1}{\beta(1 - b_1 + b_2) - mb_2}\,.$$

□

The following lemma gives a matching upper bound on the relative worst order ratio.

Lemma 7. *For any pair of values* $0 \le b_1 < b_2 \le \frac{\beta-1}{m-1}$, $WR_{\text{APRE}_{b_1}, \text{APRE}_{b_2}} \le \frac{\beta - mb_1}{\beta(1+b_2-b_1) - mb_2}$.

Proof: Consider any input sequence σ. By Lemma 4, there are three possible cases:

- If $\text{APRE}_{b_2}(\sigma) = P$, then $\text{APRE}_{b_1}(\sigma) \le \text{APRE}_{b_2}(\sigma)$.
- If $\text{APRE}_{b_2}(\sigma) = \frac{\beta}{m} P$, then by Lemma 1, $p_{\max} \le \frac{P}{m}$, and $\text{APRE}_{b_1}(\sigma) = \text{APRE}_{b_2}(\sigma)$.
- Finally, if $\text{APRE}_{b_2}(\sigma) = (\beta - mb_2)\, p_{\max} + b_2 P$, then by Lemma 1, $p_{\max} \ge \frac{P}{m}$.
  By the same lemma we get $M^t_{b_1} = (\beta - mb_1)\, p_{\max} + b_1 P$, and thus $\text{APRE}_{b_1}(\sigma) = \min\{P, (\beta - mb_1)\, p_{\max} + b_1 P\}$. If $\text{APRE}_{b_1}(\sigma) = P$, we have $\frac{P}{p_{\max}} \le \frac{\beta - mb_1}{1-b_1}$. Consequently,

$$\frac{\text{APRE}_{b_1}(\sigma)}{\text{APRE}_{b_2}(\sigma)} = \frac{P}{(\beta - mb_2)\, p_{\max} + b_2 P} \le \frac{P}{(\beta - mb_2)\, P \frac{1-b_1}{\beta - mb_1} + b_2 P}$$
$$= \frac{\beta - mb_1}{\beta(1 - b_1 + b_2) - mb_2}.$$

Otherwise, $\frac{P}{p_{\max}} \ge \frac{\beta - mb_1}{1-b_1}$ and $\text{APRE}_{b_1}(\sigma) = (\beta - mb_1)\, p_{\max} + b_1 P$. We have,

$$\frac{\text{APRE}_{b_1}(\sigma)}{\text{APRE}_{b_2}(\sigma)} = \frac{(\beta - mb_1)\, p_{\max} + b_1 P}{(\beta - mb_2)\, p_{\max} + b_2 P} = \frac{(\beta - mb_1) + b_1 \frac{P}{p_{\max}}}{(\beta - mb_2) + b_2 \frac{P}{p_{\max}}}\,.$$

This is a function which is monotonically decreasing in $\frac{P}{p_{\max}}$. Thus, we can substitute $\frac{P}{p_{\max}} = \frac{\beta - mb_1}{1-b_1}$ to find its maximum, which is again $\frac{\beta - mb_1}{\beta(1-b_1+b_2) - mb_2}$.

□

As the ratios proved in the two lemmas match, we arrive at the following theorem.

Theorem 2. *For any pair of values* $0 \le b_1 < b_2 \le \frac{\beta-1}{m-1}$,

$$WR_{\mathrm{APRE}_{b_1},\mathrm{APRE}_{b_2}} = \frac{\beta - mb_1}{\beta(1+b_2-b_1)-mb_2} > 1\ ,$$

where $\beta = \frac{\theta^m}{\theta^m-1}$ *and* $\theta = \frac{m}{m-1}$.

Proof: The ratio follows directly from Lemmas 6 and 7. It is easily checked that the ratio is greater than 1:

$$\begin{aligned} &\frac{\beta - mb_1}{\beta(1+b_2-b_1)-mb_2} > 1 \\ \Leftrightarrow\ &\beta - mb_1 > \beta(1+b_2-b_1)-mb_2, \quad \text{since } \beta(1+b_2-b_1)-mb_2 > 0 \\ \Leftrightarrow\ &m(b_2-b_1) > \beta(b_2-b_1) \end{aligned}$$

□

Substituting $b_1 = 0$ and $b_2 = \frac{\beta-1}{m-1}$ in $\frac{\beta-mb_1}{\beta(1-b_1+b_2)-mb_2}$, we get the following corollary.

Corollary 1. $WR_{\mathrm{PRE,MPRE}} = \beta\ \frac{m-1}{m+\beta^2-2\beta}$, *where* $\beta = \frac{\theta^m}{\theta^m-1}$ *and* $\theta = \frac{m}{m-1}$.

For $m = 2$, 3, and 4, the ratio $\mathrm{WR}_{\mathrm{PRE,MPRE}}$ is $\frac{6}{5} = 1.2$, $\frac{171}{131} \approx 1.365$, and $\frac{11200}{8203} \approx 1.466$, respectively. As m approaches infinity, the ratio approaches β, the competitive ratio of the two algorithms. Note that for any other pair of algorithms considered in this section, the relative worst order ratio is smaller.

2.2 Two related machines

We now turn to the case of two uniformly related machines, m_1 of speed 1 and m_2 of speed $s \ge 1$. As in the previous section, the goal is to minimize the makespan, and preemption is allowed.

For any input sequence σ, a straightforward optimal offline algorithm was found by Gonzalez and Sahni (1978). The makespan found by this algorithm is

$$\mathrm{OPT}(\sigma) = \max\left\{\frac{P}{s+1}, \frac{p_{\max}}{s}\right\}.$$

Two algorithms with optimal competitive ratio. Two slightly different deterministic online algorithms found independently by Wen and Du (1998), and by Epstein et al. (2001) both have an optimal competitive ratio of

$$\mathrm{CR} = \alpha = \frac{(s+1)^2}{s^2+s+1} = 1 + \frac{s}{s^2+s+1}.$$

This is $\frac{4}{3}$ for $s = 1$, and decreases for increasing s. The ratio is optimal even if randomization is allowed.

The algorithm of Wen and Du (1998) works similarly to the algorithm PRE (Chen et al., 1995). For job J_{t+1}, it reserves the time interval $[\frac{L'_2}{s}, \alpha\,\mathrm{OPT}^{t+1}]$ on the fast machine, and the

interval $[L_1^t, \frac{L_2^t}{s}]$ on the slow machine. Each job is assigned first to the reserved interval on the fast machine, and the remainder if any, to the other reserved interval.

The algorithm of Epstein et al. (2001) is different in the sense that on assignment of a job, it always assigns to the slow machine as much as possible, but not more than $\frac{P^{t+1}}{s^2+s+1}$, and not more than $\frac{L_2^t}{s}$ (to avoid overlap). The remainder of the job is assigned to the fast machine.

A class of algorithms with optimal competitive ratio. We define a class of algorithms called TWO-PREEMPTIVE$_c$ (TPRE$_c$) which use a parameter c, such that $0 \le c \le c_{\max}$, $c_{\max} = \frac{s^2}{s^2+s+1}$. Whenever a new job J^{t+1} arrives, schedule as large a fraction as possible on the fast machine within the time interval $[\frac{1}{s}L_2^t, \frac{1}{s}M_c^{t+1}]$, where

$$M_c^{t+1} = \max\left\{\alpha\,\frac{s}{s+1}\,P^{t+1},\ \ p_{\max}^{t+1}\left(1+\frac{c}{s}\right)+\left(P^{t+1}-p_{\max}^{t+1}\right)(c_{\max}-c)\right\}.$$

Schedule the remaining part of J_{t+1} on the slow machine, within the time interval $[L_1^t, \frac{L_2^t}{s}]$.

In some cases, TPRE$_c$ achieves a makespan which is better than that of the algorithm of Wen and Du, (1998). Later, we also compare it to the algorithm of Epstein et al. (2001). First, we prove that the algorithm is well-defined and has an optimal competitive ratio of α (Theorem 3). For that we need the following lemma.

Lemma 8. *After t jobs,* TPRE$_c$ *has* $M_c^t = \alpha\,\frac{s}{s+1}\,P^t$ *if and only if* $P^t \ge \frac{s+1}{s}\,p_{\max}^t$.

Proof: This follows from

$$\begin{aligned}
&\frac{s(s+1)}{s^2+s+1}P^t \ \ge\ p_{\max}^t\left(1+\frac{c}{s}\right)+\left(P^t-p_{\max}^t\right)\left(\frac{s^2}{s^2+s+1}-c\right)\\
\Leftrightarrow\quad &\frac{s+c(s^2+s+1)}{s^2+s+1}P^t \ \ge\ \left(\frac{s+1}{s^2+s+1}+c\,\frac{s+1}{s}\right)p_{\max}^t\\
\Leftrightarrow\quad &(s+c(s^2+s+1))P^t \ \ge\ \frac{s+1}{s}(s+c(s^2+s+1))p_{\max}^t\\
\Leftrightarrow\quad &P^t \ \ge\ \frac{s+1}{s}p_{\max}^t
\end{aligned}$$

□

Theorem 3. *For any c,* $0 \le c \le c_{max}$, TPRE$_c$ *is well-defined and has competitive ratio* α.

Proof: First, we prove that the algorithm is well-defined, i.e., the reserved time intervals are always sufficiently long. To this end we show that the algorithm maintains the following two invariants.

1. $L_2^t \le M_c^t$
2. $L_1^t \le \dfrac{P^t}{s^2+s+1}$

We consider the jobs to be scheduled one at a time, and show that each job can be scheduled observing the two invariants. For J_1, the invariants clearly hold, since the job is completely scheduled on the fast machine, and $M_c^1 = p_1(1+\frac{c}{s}) \ge p_1$.

Now consider J_{t+1}, $t \ge 1$, and assume that the invariants hold just before J_{t+1} is assigned. By the definition of the algorithm, the first invariant still holds after assigning

J_{t+1}. As for the second invariant, if $L_1^{t+1} = L_1^t$ then it is clearly maintained. Otherwise $L_1^{t+1} \le P^{t+1} - M_c^{t+1} \le P^{t+1} - \alpha \frac{s P^{t+1}}{s+1} = \frac{P^{t+1}}{s^2+s+1}$. This is exactly the second invariant.

The amount of time available for the new job is

$$\left(M_c^{t+1} - L_2^t\right) + \left(\frac{L_2^t}{s} - L_1^t\right) = M_c^{t+1} + \frac{L_2^t}{s} - P^t.$$

To complete the proof that the algorithm is well-defined, we just need to prove that this amount is at least p_{t+1}, or equivalently, that

$$M_c^{t+1} \ge -\frac{L_2^t}{s} + P^t + p_{t+1} = \frac{L_1^t - P^t}{s} + P^t + p_{t+1} = \frac{L_1^t}{s} + P^{t+1} - \frac{P^t}{s}.$$

We consider two cases depending on which of the two terms in the definition of M_c^{t+1} is maximum.

– If $M_c^{t+1} = \alpha \frac{s P^{t+1}}{s+1}$ then By Lemma 8, $P^{t+1} \ge \frac{s+1}{s} p_{\max}^{t+1}$. Thus,

$$P^t = P^{t+1} - p_{t+1} \;\ge\; P^{t+1} - p_{\max}^{t+1} \;\ge\; \frac{1}{s}\, p_{\max}^{t+1} \;\ge\; \frac{1}{s}\, p_{t+1}.$$

Further, by using the invariant $L_1^t \le \frac{P^t}{s^2+s+1}$ it suffices to prove

$$M_c^{t+1} = \alpha \frac{s P^{t+1}}{s+1} = \frac{s^2+s}{s^2+s+1}(P^t + p_{t+1}) \;\ge\; \frac{P^t}{s(s^2+s+1)} + P^{t+1} - \frac{P^t}{s}.$$

By rearranging the terms and using $P^{t+1} = P^t + p_{t+1}$, we get

$$\frac{s}{s^2+s+1}\, P^t \;\ge\; \frac{1}{s^2+s+1} p_{t+1},$$

which follows directly from $P^t \ge \frac{1}{s} p_{t+1}$.

– In the second case, by the definition of M_c^{t+1} we need to show

$$M_c^{t+1} = p_{\max}^{t+1}\left(1 + \frac{c}{s}\right) + \left(P^{t+1} - p_{\max}^{t+1}\right)\left(\frac{s^2}{s^2+s+1} - c\right) \;\ge\; \frac{L_1^t}{s} + P^{t+1} - \frac{P^t}{s}.$$

Rearranging we get

$$\left(\frac{s+1}{s^2+s+1} + \frac{s+1}{s}c\right) p_{\max}^{t+1} \ge \frac{L_1^t}{s} + \left(\frac{s+1}{s^2+s+1} + c\right) P^{t+1} - \frac{P^t}{s}.$$

By Lemma 8, $p_{\max}^{t+1} \ge \frac{s}{s+1} P^{t+1}$. Substituting $\frac{s}{s+1} P^{t+1}$ for $p_{\max}^{t+1}$ on the left hand side, we get $\frac{P^t}{s(s^2+s+1)} \ge \frac{L_1^t}{s}$, which is implied immediately by the second invariant.

Next, we show that $\text{TPRE}_c(\sigma) \le \alpha\, \text{OPT}(\sigma)$. This is clear in the first case. In the second case, $P \le \frac{s+1}{s} p_{\max}$, i.e., $P - p_{\max} \le \frac{p_{\max}}{s}$. Thus,

$$\begin{aligned}
\text{TPRE}_c(\sigma) \;&\le\; \frac{p_{\max}}{s}\left(1 + \frac{c}{s}\right) + \frac{p_{\max}}{s^2}\left(\frac{s^2}{s^2+s+1} - c\right) \\
&=\; \frac{p_{\max}}{s}\left(1 + \frac{c}{s} + \frac{s}{s^2+s+1} - \frac{c}{s}\right) = \alpha\, \frac{p_{\max}}{s} \;\le\; \alpha\, \text{OPT}(\sigma).
\end{aligned}$$

□

We first establish the relative worst order ratio between pairs of algorithms in the class TPRE_c. Similarly to Lemma 4, we can show the following lemma in which we identify the

properties of the outputs of the algorithms defined above and find an order which is always a worst ordering.

Lemma 9. *For* TPRE_c *and for every c, and any input sequence* σ*, a non-increasing order is always a worst order. The makespan for a worst permutation,* σ_w*, of* σ *is*

$$\text{TPRE}_c(\sigma_w) = \min\left\{\frac{P}{s}, \frac{1}{s}M_c(\sigma)\right\}.$$

Proof: Note that no order can give a larger makespan, so it is enough to show that a non-increasing order actually gives this makespan. Let σ be any input sequence, where the job sizes are given in non-increasing order, i.e., $p_1 = p_{\max}$ and $p_{i-1} \geq p_i$ for $i \geq 2$. Note that $p^t_{\max} = p_{\max} = p_1$ for all t.

First, we prove the following claim. If the makespan of TPRE_c after job J_t is assigned is $\frac{1}{s}M^t_c$, then the makespan after J_{t+1} is assigned must be $\frac{1}{s}M^{t+1}_c$. We have two cases:

- If $M^{t+1}_c = \alpha\frac{sP^{t+1}}{s+1}$, then using $M^t_c \geq \alpha\frac{sP^t}{s+1}$, we get that the total load that can be scheduled within the designated interval on the fast machine for J_{t+1} is at most

$$M^{t+1}_c - M^t_c \;\leq\; \alpha\frac{sP^{t+1}}{s+1} - \alpha\frac{sP^t}{s+1} = \alpha\frac{s}{s+1}p_{t+1} = \frac{s^2+s}{s^2+s+1}p_{t+1} \;<\; p_{t+1}\,,$$

  i.e., the interval is filled completely.
- If $M^{t+1}_c = p^{t+1}_{\max}(1+\frac{c}{s}) + (P^{t+1} - p^{t+1}_{\max})(c_{\max} - c)$, then the total load that can be scheduled within the designated interval on the fast machine for J_{t+1} is

$$M^{t+1}_c - M^t_c \;\leq\; p_{t+1}\,(c_{\max} - c) \;<\; p_{t+1}\,,$$

  i.e., the interval is again filled completely. The claim is thereby proved.

We are now ready to prove the lemma. As long as jobs are assigned so that the designated interval on the fast machine is not filled, there are no jobs assigned to the slow machine, and the makespan is $\frac{P^t}{s}$ after job J_t. Once this interval is filled completely, we showed that it will be filled in the next steps as well. This proves the lemma. □

Lemma 10. *For every pair of values* $0 \leq c_1 < c_2 \leq c_{max}$*,* TPRE_{c_1} *and* TPRE_{c_2} *are comparable, and* TPRE_{c_1} *is never worse than* TPRE_{c_2}*.*

Proof: Consider any input sequence σ. By Lemma 9, we only need to check the relation between the makespans of the two algorithms for a non-increasing sorted order of jobs with makespan as shown in the Lemma.

We claim that TPRE_{c_2} is never better than TPRE_{c_1} according to the relative worst order ratio. By Lemma 9, there are three cases:

- If $\text{TPRE}_{c_2}(\sigma) = \frac{P}{s}$, then clearly $\text{TPRE}_{c_1}(\sigma) \leq \text{TPRE}_{c_2}(\sigma)$.
- If $\text{TPRE}_{c_2}(\sigma) = \alpha\frac{P}{s+1}$, then by Lemma 8, $p_{\max} \leq \frac{sP}{s+1}$ and thus, $\text{TPRE}_{c_1}(\sigma) \leq \frac{M_c(\sigma)}{s} = \alpha\frac{P}{s+1} = \text{TPRE}_{c_2}(\sigma)$.
- If the makespan of TPRE_{c_2} is given by the second term in the maximum, then by Lemma 8, $p_{\max} \geq \frac{sP}{s+1}$. Therefore the makespan of TPRE_{c_1} can either be given by the second term in the maximum, or it can be less (i.e., equal to $\frac{P}{s}$). The second term in the maximum is a

function which is monotonically non-decreasing as a function of c for the case $p_{\max} \geq \frac{sP}{s+1}$, therefore the makespan of TPRE_{c_1} is not larger than the one of TPRE_{c_2}. □

Lemma 11. *For any pair of values* $0 \leq c_1 < c_2 \leq c_{max}$,

$$WR_{\text{TPRE}_{c_2},\text{TPRE}_{c_1}} \geq \frac{c_2(s^2+s+1)+s}{c_2(s^2+s)+c_1+s}.$$

Proof: Consider the input sequence $\sigma = \langle s^2+s+s(s^2+s+1)c_2,\ (s^2+s+1)c_2 \rangle$. By Lemma 9, it is enough to consider this permutation of the sequence.

We have $P = s^2+s+c_2(s+1)(s^2+s+1)$, $p_{\max} = s^2+s+c_2 \cdot s(s^2+s+1)$, and $\frac{s+1}{s}\, p_{\max} = (s+1)(s+1+c_2(s^2+s+1)) > P$. Thus, by Lemma 8,

$$\begin{aligned} M_{c_2} &= p_{\max}\left(1+\frac{c_2}{s}\right) + (P-p_{\max})\left(\frac{s^2}{s^2+s+1} - c_2\right) \\ &= s^2+s+c_2 \cdot s(s^2+s+1) + c_2(s+1+c_2(s^2+s+1)) + c_2 s^2 - c_2^2(s^2+s+1) \\ &= s^2+s+c_2(s+1)(s^2+s+1) = P. \end{aligned}$$

This function is monotonically increasing as a function of c_2, hence $M_{c_1} < M_{c_2} = P$. Thus, a lower bound on $\text{WR}_{\text{TPRE}_{c_2},\text{TPRE}_{c_1}}$ is given by $\frac{M_{c_2}}{M_{c_1}}$. We have

$$\begin{aligned} M_{c_1} &= p_{\max}\left(1+\frac{c_1}{s}\right) + (P-p_{\max})\left(\frac{s^2}{s^2+s+1} - c_1\right) \\ &= s^2+s+c_2 \cdot s(s^2+s+1) + c_1(s+1+c_2(s^2+s+1)) + c_2 s^2 - c_1 c_2(s^2+s+1) \\ &= s^2+s+c_2 s(s^2+2s+1) + c_1(s+1) \\ &= (s+1)\left(s+c_2 s(s+1)+c_1\right). \end{aligned}$$

This gives the stated lower bound. □

Lemma 12. *For any pair of values* $0 \leq c_1 < c_2 \leq c_{max}$,

$$WR_{\text{TPRE}_{c_2},\text{TPRE}_{c_1}} \leq \frac{c_2(s^2+s+1)+s}{c_2(s^2+s)+c_1+s}.$$

Proof: Consider any input sequence σ. By Lemma 9, we only need to consider the sequence in non-increasing size order. By Lemma 9, there are three possible cases:

- If $\text{TPRE}_{c_1}(\sigma) = \frac{P}{s}$, then $\text{TPRE}_{c_2}(\sigma) \leq \text{TPRE}_{c_1}(\sigma)$ (and thus, by Lemma 10, $\text{TPRE}_{c_2}(\sigma) = \text{TPRE}_{c_1}(\sigma)$).
- If $\text{TPRE}_{c_1}(\sigma) = \alpha\frac{P}{s+1}$, then by Lemma 8, $p_{\max} \leq \frac{sP}{s+1}$, and $\text{TPRE}_{c_2}(\sigma) = \text{TPRE}_{c_1}(\sigma)$.
- Finally, if the makespan of TPRE_{c_1} is given by the second term in the maximum (with the parameter c_1), by Lemma 8, $P \leq \frac{s+1}{s}\, p_{\max}$, and thus by the same lemma, the makespan of TPRE_{c_2} is either given by the second term in the maximum (with the parameter c_2) or equal to $\frac{P}{s}$.

– If $\mathrm{TPRE}_{c_2}(\sigma) = \frac{P}{s}$, we get

$$\mathrm{WR}_{\mathrm{TPRE}_{c_2},\mathrm{TPRE}_{c_1}} = \frac{P}{p_{\max}\left(1+\frac{c_1}{s}\right)+(P-p_{\max})(c_{\max}-c_1)} = \frac{\frac{P}{p_{\max}}}{\left(1+\frac{c_1}{s}\right)+\left(\frac{P}{p_{\max}}-1\right)(c_{\max}-c_1)}.$$

This function is monotonically non-decreasing in the ratio $\frac{P}{p_{\max}}$, and thus we calculate the maximum ratio. Since $\mathrm{TPRE}_{c_2}(\sigma) = \frac{P}{s}$, $P \le p_{\max}(1+\frac{c_2}{s}) + (P-p_{\max})(\frac{s^2}{s^2+s+1} - c_2)$, which is equivalent to

$$P\left(c_2 + \frac{s+1}{s^2+s+1}\right) \le p_{\max}\left(c_2\left(1+\frac{1}{s}\right) + \frac{s+1}{s^2+s+1}\right).$$

Thus,

$$\frac{P}{p_{\max}} \le \frac{c_2\left(1+\frac{1}{s}\right)+\frac{s+1}{s^2+s+1}}{c_2+\frac{s+1}{s^2+s+1}} = \frac{c_2(s^2+s+1)+s}{c_2(s^2+s+1)\frac{s}{s+1}+s} = \frac{x+s}{\frac{s}{s+1}x+s},$$

where $x = c_2(s^2+s+1)$. Substituting this in the upper bound on $\mathrm{WR}_{\mathrm{TPRE}_{c_2},\mathrm{TPRE}_{c_1}}$ we get

$$\mathrm{WR}_{\mathrm{TPRE}_{c_2},\mathrm{TPRE}_{c_1}} \le \frac{\frac{x+s}{\frac{s}{s+1}x+s}}{\left(1+\frac{c_1}{s}\right)+\left(\frac{x+s}{\frac{s}{s+1}x+s}-\frac{\frac{s}{s+1}x+s}{\frac{s}{s+1}x+s}\right)(c_{\max}-c_1)} = \frac{x+s}{\left(1+\frac{c_1}{s}\right)\left(\frac{s}{s+1}x+s\right)+\left(x-\frac{s}{s+1}x\right)(c_{\max}-c_1)} = \frac{x+s}{N(x)},$$

where

$$\begin{aligned} N(x) &= \frac{s}{s+1}x + s + \left(\frac{1}{s+1}x + 1 - x + \frac{s}{s+1}x\right)c_1 + \left(x - \frac{s}{s+1}x\right)c_{\max} \\ &= \frac{s}{s+1}x + s + c_1 + xc_{\max} - \frac{s}{s+1}xc_{\max} \\ &= \frac{s}{s+1}x(1-c_{\max}) + s + c_1 + xc_{\max} \\ &= \frac{s}{s+1}c_2(s^2+s+1)\left(1-\frac{s^2}{s^2+s+1}\right) + s + c_1 + c_2(s^2+s+1)\frac{s^2}{s^2+s+1} \\ &= c_2 s + s + c_1 + c_2 s^2 \end{aligned}$$

This gives the stated upper bound.

– If $\text{TPRE}_{c_2}(\sigma) < \frac{P}{s}$, we get

$$\begin{aligned}\text{WR}_{\text{TPRE}_{c_2},\text{TPRE}_{c_1}} &= \frac{p_{\max}\left(1+\frac{c_2}{s}\right)+(P-p_{\max})(c_{\max}-c_2)}{p_{\max}\left(1+\frac{c_1}{s}\right)+(P-p_{\max})(c_{\max}-c_1)} \\ &= \frac{\left(1+\frac{c_2}{s}\right)+\left(\frac{P}{p_{\max}}-1\right)(c_{\max}-c_2)}{\left(1+\frac{c_1}{s}\right)+\left(\frac{P}{p_{\max}}-1\right)(c_{\max}-c_1)}.\end{aligned}$$

This function is monotonically non-increasing as a function of the ratio $\frac{P}{p_{\max}}$, and thus we calculate the maximum ratio. Since $\text{TPRE}_{c_2}(\sigma) < \frac{P}{s}$, we get $P \geq p_{\max}(1+\frac{c_2}{s})+(P-p_{\max})(\frac{s^2}{s^2+s+1}-c_2)$. Thus, similarly to the previous case we get

$$\frac{P}{p_{\max}} \geq \frac{c_2\left(s^2+s+1\right)+s}{c_2\left(s^2+s+1\right)\frac{s}{s+1}+s} = 1+\frac{1}{s}\,\frac{c_2(s^2+s+1)}{c_2(s^2+s+1)+s+1},$$

Substituting this in the upper bound on $\text{WR}_{\text{TPRE}_{c_2},\text{TPRE}_{c_1}}$ we get (using the definition of $c_{\max}$ and simple algebra),

$$\begin{aligned}\text{WR}_{\text{TPRE}_{c_2},\text{TPRE}_{c_1}} &\leq \frac{(s+c_2)\left(c_2(s^2+s+1)+s+1\right)+c_2(s^2+s+1)(c_{\max}-c_2)}{(s+c_1)\left(c_2(s^2+s+1)+s+1\right)+c_2(s^2+s+1)(c_{\max}-c_1)} \\ &= \frac{s\left(c_2(s^2+s+1)+s+1\right)+c_2(s+1)+c_2s^2}{s\left(c_2(s^2+s+1)+s+1\right)+c_1(s+1)+c_2s^2} \\ &= \frac{c_2(s+1)(s^2+s+1)+s^2+s}{c_2(s^3+s^2+s)+c_1(s+1)+s^2+s} \\ &= \frac{c_2(s^2+s+1)+s}{c_2(s^2+s)+c_1+s}.\end{aligned}$$

□

Theorem 4. *For any pair of values* $0 \leq c_1 < c_2 \leq c_{max}$, $WR_{\text{TPRE}_{c_2},\text{TPRE}_{c_1}} = \frac{c_2(s^2+s+1)+s}{c_2(s^2+s)+c_1+s}$.

By substituting $c_1 = 0$ and $c_2 = c_{\max}$ we get the following corollary.

Corollary 2. $WR_{\text{TPRE}_{c_{max}},\text{TPRE}_0} \geq 1+\frac{s}{s^3+2s^2+s+1}$.

Next, we would like to compare the above class of algorithms to the algorithm of Epstein et al. (2001) called BEST PREEMPTIVE (BPRE). We first show that the best algorithm in the class, TPRE_0, is worse than BPRE. Due to the transitivity of the relative worst order ratio, this implies that BPRE is better than all algorithms TPRE_c for any c.

The algorithm BPRE is different from the above algorithms in the sense that it, except for the first job which is scheduled completely on the fast machine, BPRE always schedules as large a part on the slow machine as possible while maintaining the following two invariants:

1. $L_1^t \leq \frac{L_2^t}{s^2+s}$.
2. If $L_1^t < \frac{L_2^t}{s^2+s}$, then also $L_2^t \leq \alpha\ p_{\max}^t$.

Note that the first invariant implies that $L_1^t \leq \frac{L_2^t}{s}$, i.e., the makespan of BPRE is always $\frac{L_2^t}{s}$. Also note that by the same invariant, $P^t = L_1^t + L_2^t \leq (\frac{1}{s^2+s} + 1)L_2^t = \frac{s^2+s+1}{s^2+s} L_2^t$, i.e.,

$$\frac{s+1}{s^2+s+1} P^t \leq \frac{1}{s} L_2^t. \tag{2.1}$$

We prove that TPRE_0 can never have a smaller makespan than BPRE, on any sequence. Note that BPRE does not necessarily have the worst makespan in the case that the sequence is sorted by non-increasing job size. As an example, consider the case $s = 2$ and the jobs 2, 12. In the order 12, 2, the job 12 is scheduled on the fast machine. The job 2 fits perfectly on the slow machine, and the makespan is $\frac{12}{2} = 6$. However, if the job 2 is assigned first, then the slow machine can receive at most size 1 in the next step, since otherwise the two parts of the second job scheduled on the slow and the fast machine, respectively, will overlap in time. Therefore, the makespan is $\frac{2+11}{2} = 6\frac{1}{2}$.

Lemma 13. *For any input sequence* σ, $\text{BPRE}(\sigma) \leq \text{TPRE}_0(\sigma)$.

Note that the claim of the lemma is stated for any sequence without any reordering.

Proof: Given a sequence of jobs, σ, we show that the lemma holds at every step using induction. Clearly, the lemma holds before any job is assigned.

Next, assume that the lemma holds for the previous part of the input sequence, and consider the next job, J_{t+1}. We now have several cases:

- If the job is assigned such that L_2^{t+1} of BPRE does not change, then the lemma holds by induction.
- If TPRE_0 assigns the job completely to the fast machine, then since BPRE may assign at most that much to the fast machine, and has a makespan which is not larger than the one of TPRE_0 before the assignment, this situation remains after the assignment.
- Invariant 2 for TPRE_0, $L_1^{t+1} \leq \frac{1}{s^2+s+1} P^{t+1}$, can be rewritten to $L_1^{t+1} \leq \frac{1}{s^2+s} L_2^{t+1}$. Hence, if BPRE assigns the job such that $L_1^{t+1} = \frac{1}{s^2+s} L_2^{t+1}$, TPRE_0 must have at least the same value of L_2^{t+1} as BPRE, i.e., the makespan of TPRE_0 is at least the same as the makespan of BPRE.
- Finally, the last case is when for BPRE $L_1^{t+1} < \frac{1}{s^2+s} L_2^{t+1}$ and it was impossible for BPRE to put everything on the slow machine, i.e., $L_1^{t+1} = \frac{1}{s} L_2^t$. Putting this together we obtain $L_2^t < \frac{1}{s+1} L_2^{t+1}$, and subsequently $P^{t+1} = L_2^{t+1} + L_1^{t+1} > (s+1)L_2^t + \frac{1}{s} L_2^t = \frac{s^2+s+1}{s} L_2^t$. Using this we get that $p_{t+1} = P^{t+1} - P^t > \frac{s^2+s+1}{s} L_2^t - P^t \geq (s+1)P^t - P^t = sP^t$, where the second inequality holds by (2.1). We conclude that $p_{t+1} = p_{\max}^{t+1}$. By Lemmas 8 and 9 and since we know that TPRE_0 did not schedule job p_{t+1} on the fast machine only, the makespan of TPRE_0 must be $M_0^{t+1}(\sigma) = \frac{1}{s} p_{t+1} + \frac{s}{s^2+s+1} P^t$.
  For BPRE by the Invariant 1, the makespan is at most

$$\frac{L_2^{t+1}}{s} = \frac{P^t + p_{t+1} - \frac{1}{s} L_2^t}{s} \leq \frac{P^t + p_{t+1} - \frac{s+1}{s^2+s+1} P^t}{s} = \frac{p_{t+1}}{s} + \frac{sP^t}{s^2+s+1}.$$

Therefore, the claim is proved. □

By Lemma 9 and since $M_c(\sigma)$ is non-decreasing as a function of c, we immediately obtain the following corollary.

Corollary 3. *For every c and any input sequence σ,* $\text{BPRE}(\sigma) \leq \text{TPRE}_c(\sigma)$.

Next, we establish the relative worst order ratio between BPRE and each algorithm TPRE_c. We prove the following theorem.

Theorem 5. *The relative worst order ratio between* BPRE *and* TPRE_c *is*

$$WR_{\text{TPRE}_c,\text{BPRE}} = \max\left\{\frac{c(s^4+2s^3+2s^2+s)+s^3+s^2}{c(s^4+2s^3+s^2-1)+s^3+s^2}, \frac{(s^2+s+1)(s^3+s+c)+s^4}{s^2(s^2+s+1)(s+1)}\right\}.$$

For $s=1$ and $c=c_{max}=\frac{s^2}{s^2+s+1}=\frac{1}{3}$, this value is $\alpha=\frac{4}{3}$. For $s>1$, this value is smaller than α. For all values of s the relative worst order ratio is a strictly monotonically increasing function of s.

Proof: To prove the lower bound, consider two input sequences.

– The first sequence consists of two jobs of sizes s and s^2+1. Here $P=s^2+s+1$ and $p_{\max}=s^2+1$. Since $M_c(\sigma)\leq P$, and since $P\leq\frac{s+1}{s}p_{\max}$, and using Lemma 8, the makespan of TPRE_c is $\frac{M_c(\sigma)}{s}=(s^2+1)(\frac{1}{s}+\frac{c}{s^2})+\frac{s^2}{s^2+s+1}-c$.
Next, we claim that the makespan of BPRE is $s+1$ for both possible permutations of the jobs in the sequence. Both cases result in the same makespan since in both cases we have a similar situation as follows. The first job is assigned to the fast machine, and out of the second job, a part of size 1 is assigned to the slow machine, to get a balance between the loads of the two machines as the the definition of the algorithm states. In total, we get the ratio

$$\frac{(s^2+s+1)(s^3+s+c)+s^4}{s^2(s^2+s+1)(s+1)}.$$

– The second sequence is defined for $c>0$. It consists of the two jobs $c(s^2+s+1)$ and $sc(s^2+s+1)+s^2+s$. Here $P=c(s+1)(s^2+s+1)+s^2+s$ and $p_{\max}=sc(s^2+s+1)+s^2+s$. Since $M_c(\sigma)\leq P$, using Lemma 8 we need to consider the second term in the maximum to find $M_c(\sigma)$. We have $M_c(\sigma)=c(s^3+2s^2+2s+1)+s^2+1=P$. Therefore, the makespan of TPRE_c is $\frac{M_c(\sigma)}{s}=\frac{P}{s}=c(s^2+2s+2+\frac{1}{s})+s+1$.
For BPRE, if the smaller job is assigned first, then the part of the second job that can be assigned on the slow machine is at most $c(s+1+\frac{1}{s})$ which is consistent with Invariant 1. If the larger job is assigned first, the part assigned to the slow machine can only be larger. Thus we get that the makespan of BPRE is at most $\frac{L_2}{s}=\frac{P}{s}-\frac{c(s^2+s+1)}{s^2}=c(s^2+2s+1-\frac{1}{s^2})+s+1$. By dividing the two, we get the ratio

$$\frac{c(s^4+2s^3+2s^2+s)+s^3+s^2}{c(s^4+2s^3+s^2-1)+s^3+s^2}.$$

To prove the upper bound, we start with giving lower bounds on the makespan of BPRE. Given an input sequence σ let $R = P - p_{\max}$, i.e., the sum of all jobs except the largest. By Invariant 1, we have $\mathrm{BPRE_W}(\sigma) \geq \frac{s+1}{s^2+s+1}(R + p_{\max})$.

Now consider an ordering of the jobs, such that $p_{\max}$ is last. Let μ be the makespan before $p_{\max}$ is assigned. After $p_{\max}$ is assigned the makespan is at least μ plus the time to run $p_{\max}$ minus the amount that can be scheduled on the slow machine, which is at most $\mu - (R - s\mu)$, i.e., the makespan is at least $\mu + \frac{p_{\max}-\mu+R-s\mu}{s} = \frac{p_{\max}-\mu+R}{s}$. Since $\mu \leq \frac{R}{s}$, we get at least $\mathrm{BPRE_W}(\sigma) \geq \frac{sp_{\max}+sR-R}{s^2}$.

The makespan of TPRE_c (for any possible order) never exceeds

$$\max\left\{\alpha\frac{P}{s+1},\, p_{\max}\left(\frac{1}{s}+\frac{c}{s^2}\right)+R\left(\frac{s}{s^2+s+1}-\frac{c}{s}\right)\right\}.$$

If the first option is the maximum then BPRE has at least the same makespan by (2.1). For the second option, we consider several cases.

- If $p_{\max} \leq R(s+\frac{1}{s})$, we use the first lower bound, and get a ratio between the two algorithms of at most

$$\frac{p_{\max}\left(\frac{1}{s}+\frac{c}{s^2}\right)+R\left(\frac{s}{s^2+s+1}-\frac{c}{s}\right)}{\frac{s+1}{s^2+s+1}(R+p_{\max})}.$$

  Let $\rho = \frac{p_{\max}}{R}$. If $R = 0$, we have $P = p_{\max}$ and thus there is only one job. In this case all algorithms act in the same way, therefore we do not consider this option. Dividing the numerator and the denominator by R and substituting we get the function $\frac{\rho(\frac{1}{s}+\frac{c}{s^2})+\frac{s}{s^2+s+1}-\frac{c}{s}}{\frac{s+1}{s^2+s+1}(1+\rho)}$. This function is monotonically non-decreasing as a function of ρ. Hence, we substitute $p_{\max} = R(s+\frac{1}{s})$, since this corresponds to the maximum value of ρ. Thus, we find the maximal value to be $\frac{(s^2+s+1)(s^3+s+c)+s^4}{s^2(s^2+s+1)(s+1)}$.
- If $p_{\max} \geq R(s+\frac{1}{s})$, we use the second lower bound and get a ratio between the two algorithms of at most

$$\frac{p_{\max}\left(\frac{1}{s}+\frac{c}{s^2}\right)+R\left(\frac{s}{s^2+s+1}-\frac{c}{s}\right)}{\frac{sp_{\max}+sR-R}{s^2}} = \frac{p_{\max}(s+c)+R\left(\frac{s^3}{s^2+s+1}-cs\right)}{sp_{\max}+sR-R}.$$

  Let $\rho = \frac{p_{\max}}{R}$. We get the function $\frac{\rho(s+c)+\frac{s^3}{s^2+s+1}-cs}{s\rho+s-1}$. We now have two cases depending on the value of c:

  - For $c \leq \frac{s}{(s^2+s+1)(s^2+s-1)}$, the function above is monotonically non-increasing (and otherwise monotonically non-decreasing). Therefore in this case we can substitute $p_{\max} = R(s+\frac{1}{s})$ or $\rho = s+\frac{1}{s}$ to find the maximum, which turns out to be $\frac{(s^2+s+1)(s^3+s+c)+s^4}{s^2(s^2+s+1)(s+1)}$.
  - For larger values of c, we consider first the case when $\rho \geq \frac{sc(s^2+s+1)+s^2+s}{c(s^2+s+1)}$. Note that this value is strictly larger than $1+\frac{1}{s}$ for any value of $c > 0$. The makespan of TPRE_c is also at most $\frac{p_{\max}+R}{s}$, and the ratio can be bounded by $\frac{\frac{p_{\max}+R}{s}}{\frac{sp_{\max}+sR-R}{s^2}} = \frac{sp_{\max}+sR}{sp_{\max}+sR-R}$.

Using $\rho = \frac{p_{\max}}{R}$ we get the function $\frac{s\rho+s}{s\rho+s-1}$ which is monotonically non-increasing as a function of ρ. We substitute using $\rho = \frac{sc(s^2+s+1)+s^2+s}{c(s^2+s+1)}$ to find the maximum which gives the value $\frac{c(s^4+2s^3+2s^2+s)+s^3+s^2}{c(s^4+2s^3+s^2-1)+s^3+s^2}$.
Otherwise, if $s + \frac{1}{s} \leq \rho \leq \frac{sc(s^2+s+1)+s^2+s}{c(s^2+s+1)}$, we substitute $\rho = \frac{sc(s^2+s+1)+s^2+s}{c(s^2+s+1)}$ into $\frac{\rho(s+c)+\frac{s^3}{s^2+s+1}-cs}{s\rho+s-1}$, which is monotonically non-decreasing for the current values of c, to get the maximum. This again gives $\frac{c(s^4+2s^3+2s^2+s)+s^3+s^2}{c(s^4+2s^3+s^2-1)+s^3+s^2}$. □

To find the relative worst order ratio between BPRE and $\text{TPRE}_{c_{\max}}$, which are the algorithms of Epstein et al. (2001) and Wen and Du (1998), we substitute the value of c and get the following corollary.

Corollary 4. $WR_{\text{TPRE}_{c_{max}},\text{BPRE}} = 1 + \frac{1}{s(s+2)}$. *This value is smaller than* α *for every* $s > 1$.

3 Non-preemptive scheduling to minimize makespan

For completeness, we consider non-preemptive algorithms as well.

3.1 Identical machines

First, we consider the scheduling problem for m identical machines where preemption is not allowed, i.e., a job cannot be interrupted and run on more than one machine. For this problem, a classical result was presented by Graham (1966). Graham considers the natural greedy algorithm LIST, which always schedules a job on the least loaded machine. By Graham (1966), LIST is $(2 - \frac{1}{m})$-competitive. This is optimal when $m \leq 3$ (Faigle et al., 1989).

For $m \geq 4$, this result was later improved. First by Galambos and Woeginger (1993) with an $(2 - \frac{1}{m} - \epsilon_m)$-competitive algorithm RLS. Unfortunately for m approaching infinity, ϵ_m tends to 0, i.e., for general m, this result is not better. Later Bartal et al. (1995) gave an algorithm which is 1.986-competitive, but only for at least 70 machines. Karger et al. (1996) generalized this and gave a 1.945-competitive algorithm CHASM_α. Albers (1999) improved this even further, and gave a 1.923-competitive algorithm M2. The current best result is by Fleischer and Wahl (2000). They present an algorithm MR with a competitive ratio of $1 + \sqrt{(1 + \ln 2)/2} < 1.92009$, but only for $m \geq 64$.

The currently best lower bound for the problem was established by Gormley et al. (2000) at 1.85358. This is a slight improvement of the previous lower bound by Albers (1999).

Whereas the first algorithm LIST keeps the load of all machines as close as possible to the average load, essentially all the later algorithms always keep a certain fraction of the machines sufficiently below the average load, such that they can accept a large job without violating the competitive ratio.

In this section, we show that Graham's algorithm, LIST, and the two most recent algorithms, M2 and MR, are pairwise incomparable using the relative worst order ratio. This is done using the following two input sequences: σ_1 consists of $m(m-1)$ unit sized jobs, and σ_2 consists of the same jobs as σ_1 with an additional large job of size m. Note that all the jobs of σ_1 are the same size, hence all permutations are equal. For σ_2, we only need to consider the location of the large job in the sequence.

The optimal algorithm distributes the jobs in σ_1 evenly among the m processors and gets a makespan of $m-1$. For σ_2, the large job is put on a machine by itself, and all the unit-sized jobs are put on the remaining $m-1$ machines, yielding a makespan of m.

LIST distributes the jobs in σ_1 similar to OPT with a makespan of $m-1$. For σ_2, the large job of size m is placed on one of the machines with a load of $m-1$, i.e., a total makespan of $2m-1$. This is the worst possible permutation for LIST.

For the last two algorithms, we only give a sketch of the proof and we only consider the case for m approaching infinity, since this simplifies the calculations. If necessary, the calculations can be done for any specific m with the same conclusion as a result, namely that the three algorithms are pairwise incomparable.

M2 and MR both divide the machines into two groups: the s least loaded (small) machines $(m_1, m_2, \ldots, m_s)$ and the remaining $m-s$ (large) machines $(m_{s+1}, m_{s+2}, \ldots, m_m)$. We have:

$$s_{\mathrm{M2}} = \left\lfloor \frac{m}{2} \right\rfloor \approx \frac{m}{2} \text{ and } s_{\mathrm{MR}} = m - \left\lceil \frac{5c - 2c^2 - 1}{c} m \right\rceil + 1 \approx \frac{2c^2 - 4c + 1}{c} m + 1$$

where $c = 1 + \sqrt{\frac{1+\ln 2}{2}}$, the competitive ratio of MR.

Depending on different conditions the algorithms choose between putting a new job on the least loaded small machine, m_1, or the least loaded large machine, m_{s+1}. When only considering unit-sized jobs, it can be shown that at any time the difference in load for the least and most loaded small machine is at most one. The same results hold for the large machines. It can also be shown that for both algorithms a worst ordering of σ_2 is when the large job appears as the last job, and in this case this job is placed on the least loaded machine, m_1.

For M2 and σ_1, the ratio between L_1 and L_m approaches α for m approaching infinity, where

$$\alpha = \frac{0.923 s_{\mathrm{M2}} - 0.145\, m}{0.923(m - s_{\mathrm{M2}})} \approx \frac{0.923 m/2 - 0.145\, m}{0.923(m - m/2)} = \frac{633}{923}.$$

Next, for σ_1, we get $m(m-1) \approx s_{\mathrm{M2}} L_1^{\sigma_1} + (m - s_{\mathrm{M2}}) L_m^{\sigma_1} = \frac{m}{2}(\alpha + 1) L_m^{\sigma_1}$, and hence $L_m^{\sigma_1} \approx \frac{2}{\alpha+1}(m-1) \approx \frac{923}{778}\,\mathrm{OPT}(\sigma_1) \approx 1.18638\,\mathrm{OPT}(\sigma_1)$.

For σ_2, the load of L_m can be found as

$$\begin{aligned} L_m^{\sigma_2} &= L_1^{\sigma_1} + m \approx \alpha L_m^{\sigma_1} + m \approx \frac{2\alpha}{\alpha+1}(m-1) + m \approx \frac{1411}{778} m - \frac{633}{778} \\ &= \frac{1411}{778}\mathrm{OPT}(\sigma_2) - \frac{633}{778}. \end{aligned}$$

For MR and σ_1, the ratio between L_1 and L_m approaches β for m approaching infinity, where

$$\beta = \frac{2c-3}{2(c-1)} = 1 - \frac{1}{2c-2},$$

with $c = 1 + \sqrt{\frac{1+\ln 2}{2}}$, the competitive ratio of MR.

Now, for σ_1, we have,

$$\begin{aligned} m(m-1) &\approx s_{\mathrm{MR}} L_1^{\sigma_1} + (m - s_{\mathrm{MR}}) L_m^{\sigma_1} \\ &\approx (s_{\mathrm{MR}}\beta + (m - s_{\mathrm{MR}})) L_m^{\sigma_1} \\ &= (m + s_{\mathrm{MR}}(\beta - 1))\, L_m^{\sigma_1} \\ &\approx \left(m - \left(\frac{2c^2 - 4c + 1}{c} m + 1 \right) \left(\frac{1}{2c-2} \right) \right) L_m^{\sigma_1} \\ &= \left((2c-2)m - \frac{2c^2 - 4c + 1}{c} m - 1 \right) \frac{L_m^{\sigma_1}}{2c-2} \\ &= ((2c-1)m - c) \frac{L_m^{\sigma_1}}{c(2c-2)} \end{aligned}$$

Hence, $L_m^{\sigma_1} \approx \frac{m(m-1)c(2c-2)}{(2c-1)m-c} \approx \frac{2c^2-2c}{2c-1}(m-1) = \frac{2c^2-2c}{2c-1}\,\mathrm{OPT}(\sigma_1) \approx 1.24405\,\mathrm{OPT}(\sigma_1)$.

For σ_2, the load of L_m can be found as

$$\begin{aligned} L_m^{\sigma_2} &= L_1^{\sigma_1} + m \\ &= \beta L_m^{\sigma_1} + m \\ &\approx \frac{2c-3}{2(c-1)} \frac{2c^2-2c}{2c-1} (m-1) + m \\ &= \frac{(2c-3)c}{2c-1} (m-1) + m \\ &= \frac{2c^2-c-1}{2c-1} m - \frac{(2c-3)c}{2c-1} \\ &= \frac{2c^2-c-1}{2c-1} \mathrm{OPT}(\sigma_2) - \frac{(2c-3)c}{2c-1} \\ &\approx 1.56801\,\mathrm{OPT}(\sigma_2) - 0.56801 \end{aligned}$$

The results for m approaching infinity are summarized in the following table. Recall that $\mathrm{ALG_W}(\sigma)$ denotes the makespan of ALG on the worst permutation of σ. Note that for any pair of the three algorithms, the order of the two is different for σ_1 when compared to σ_2.

ALG	$\frac{\mathrm{ALG_W}(\sigma_1)}{\mathrm{OPT}(\sigma_1)}$	$\frac{\mathrm{ALG_W}(\sigma_2)}{\mathrm{OPT}(\sigma_2)}$
LIST	1	2
M2	1.18638	1.81362
MR	1.24405	1.56801

Theorem 6. LIST, M2, *and* MR *are pairwise incomparable.*

It is not surprising that LIST is incomparable to M2 and MR. The two latter algorithms are designed to do slightly bad on some input sequences, like the sequence with unit sized jobs, in order to avoid even worse performance on other input sequences.

3.2 Two related machines

Assume now that we have two machines available, and one machine is a factor of s times faster than the other, $s \geq 1$. Preemption is still not allowed.

Let POST-GREEDY be the algorithm that schedules each job on the machine where it will finish first. By Cho and Sahni (1980), Epstein and Sgall (2000) and Epstein et al. (2001), POST-GREEDY has an optimal competitive raito of $\frac{2s+1}{s+1}$, if $s \leq \phi$, and $\frac{s+1}{s}$, if $s \geq \phi$, where $\phi \approx 1.618$ is the golden ratio. It is easy to see that the algorithm FAST that simply schedules all jobs on the fastest machine is $\frac{s+1}{s}$-competitive (Epstein et al., 2001). Hence, for $s \geq \phi$, both algorithms have the optimal competitive ratio. However, POST-GREEDY seems to be the more reasonable algorithm: it never gives a larger makespan than FAST, and in many cases it even has a much smaller makespan. This is reflected by the relative worst order ratio:

For any $n \geq 1$, consider the input sequence consisting of $\lfloor n(s+1) \rfloor$ jobs of unit size. On this input sequence, FAST has a makespan of $\left\lfloor \frac{n(s+1)}{s} \right\rfloor$ and POST-GREEDY has a makespan of at most n. Since $\left\lfloor \frac{n(s+1)}{s} \right\rfloor$ approaches $\frac{s+1}{s}$ as n approaches infinity, $\mathrm{WR}_{\text{FAST,POST-GREEDY}} \geq \frac{s+1}{s}$, and since this ratio cannot be larger than the competitive ratio of FAST, the result is tight.

Theorem 7. $WR_{\text{FAST,POST-GREEDY}} = \frac{s+1}{s}$.

4 Conclusion

In this work we have applied the relative worst order ratio to a few online problems.

For most of the considered scheduling problems, competitive analysis does not distinguish between different optimal algorithms, whereas using the relative worst order ratio we are able to distinguish the algorithms, and in all cases the ratio prefers the intuitively better algorithm.

For non-preemptive scheduling on identical machines, the considered algorithms are incomparable using the relative worst order ratio, even when the algorithms have different competitive ratios. The reason for this is that, except for the first algorithm LIST, the algorithms have been specially tailored to get good competitive ratios, i.e., to work well on worst-case sequences. This is done at the expense of getting a bad makespan for many normal input sequences, where the algorithms with a worse competitive ratio are better. The competitive ratio measure in this case prefers certain non-preemptive algorithms whereas the relative worst order ratio allows us to see that the algorithms are incomparable. The order of their relative performance depends on the type of input sequence given, and it is impossible to say that one algorithm is generally better than the other for all input sequences.

In general, our results show that, in many cases, the relative worst order ratio can motivate searching for better algorithms, even when an algorithm with optimal competitive ratio has been found. We saw that in many cases, a very small change in the algorithm, without changing the competitive ratio, can be immediately seen in the resulting relative worst order ratio.

References

Albers S (1999) Better bounds for online scheduling. SIAM J Comput 29(2):459–473

Bartal Y, Fiat A, Karloff H, Vohra R (1995) New algorithms for an ancient scheduling problem. J Comput Syst Sci 51(3):359–366

Ben-David S, Borodin A (1994) A new measure for the study of on-line algorithms. Algorithmica 11(1):73–91

Boyar J, Favrholdt LM (2003) The relative worst order ratio for on-line algorithms. In Proc. 5th Italian conf. on algorithms and complexity, vol. 2653 of Lect Notes Comp Sci Springer-Verlag, pp 58–69
Boyar J, Favrholdt LM, Larsen KS (2005) The relative worst order ratio applied to paging. In Proc. 16th Annu. ACM-SIAM symp. discrete algorithms, pp 718–727
Boyar J, Medvedev P (2004) The relative worst order ratio applied to seat reservation. In Proc. of the 9th scand. workshop on algorithm theory, vol. 3111 in Lect Notes Comp Sci pp 90–101
Chen B, van Vliet A, Woeginger GJ (1995) An optimal algorithm for preemptive on-line scheduling. Oper Res Lett 18(3):127–131
Cho Y, Sahni S (1980) Bounds for list schedules on uniform processors. SIAM J Comput 9(1):91–103
Epstein L, Noga J, Seiden SS, Sgall J, Woeginger GJ (2001) Randomized online scheduling on two uniform machines. J Sched 4(2):71–92
Epstein L, Sgall J (2000) A lower bound for on-line scheduling on uniformly related machines. Oper Res Lett 26(1):17–22
Faigle U, Kern W, Turán G (1989) On the performance of on-line algorithms for partition problems. Acta Cybernet 9(2):107–119
Fleischer R, Wahl M (2000) On-line scheduling revisited. J Sched 3(6):343–353
Galambos G, Woeginger GJ (1993) An on-line scheduling heuristic with better worst case ratio than Graham's list scheduling. SIAM J Comput 22(2):349–355
Gonzalez T, Sahni S (1978) Preemptive scheduling of uniform processor systems. J ACM 25(1):92–101
Gormley T, Reingold N, Torng E, Westbrook J (2000) Generating adversaries for request-answer games. In Proc. 11th annu. ACM-SIAM symp. on discrete algorithms, pp 564–565
Graham RL (1966) Bounds for certain multiprocessing anomalies. Bell Systems Techn J 45:1563–1581
Karger DR, Philips SJ, Torng E (1996) A better algorithm for an ancient scheduling problem. J Algorithms 20(2):400–430
Kenyon C (1996) Best-fit bin-packing with random order. In Proc. 7th annu. ACM-SIAM symp. on discrete algorithms, pp 359–364
Kohrt JS (2004) Online algorithms under new assumptions, PhD thesis, Dept Math and Comp Sci, Univ South Den, p. 78.
McNaughton R (1959) Scheduling with deadlines and loss functions. Manag Sci 6(1):1–12
Seiden SS (2001) Preemptive multiprocessor scheduling with rejection. Theoret Comp Sci 262(1–2):437–458
Wen J, Du D (1998) Preemptive on-line scheduling for two uniform processors. Oper Res Lett 23:113–116