Math. Program., Ser. A (2009) 119:109–136
DOI 10.1007/s10107-007-0204-7

FULL LENGTH PAPER

LP-based online scheduling: from single to parallel machines

José R. Correa · Michael R. Wagner

Received: 16 May 2007 / Accepted: 28 November 2007 / Published online: 4 January 2008

Abstract We study classic machine sequencing problems in an online setting. Specifically, we look at deterministic and randomized algorithms for the problem of scheduling jobs with release dates on identical parallel machines, to minimize the sum of weighted completion times: Both preemptive and non-preemptive versions of the problem are analyzed. Using linear programming techniques, borrowed from the single machine case, we are able to design a 2.62-competitive deterministic algorithm for the non-preemptive version of the problem, improving upon the 3.28-competitive algorithm of Megow and Schulz. Additionally, we show how to combine randomization techniques with the linear programming approach to obtain randomized algorithms for both versions of the problem with competitive ratio strictly smaller than 2 for any number of machines (but approaching two as the number of machines grows). Our algorithms naturally extend several approaches for single and parallel machine scheduling. We also present a brief computational study, for randomly generated problem instances, which suggests that our algorithms perform very well in practice.

Keywords Online algorithms · Machine scheduling

A preliminary version of this work appears in the Proceedings of the 11th conference on integer programming and combinatorial optimization (IPCO), Berlin, 8–10 June 2005.

J. R. Correa
School of Business, Universidad Adolfo Ibáñez, Santiago, Chile
e-mail: correa@uai.cl

M. R. Wagner (✉)
Department of Management,
California State University East Bay, Hayward, CA 94542, USA
e-mail: michael.wagner@csueastbay.edu

1 Introduction

We study online versions of classic parallel machine scheduling problems. Given a set of jobs $N = \{1, \ldots, n\}$, where each job j has a processing time $p_j > 0$, a weight $w_j > 0$ and a release date $r_j \geq 0$, we want to process these jobs on m identical machines. We consider both non-preemptive and preemptive versions; in the latter case, a job being processed may be interrupted and resumed later, possibly on a different machine. Letting C_j be the completion time of job j in a given schedule, we are interested in minimizing the weighted sum of completion times: $\sum_{j \in N} w_j C_j$. As we only consider online problems, jobs are not known until their respective release date. In the scheduling notation of Graham et al. [12], we consider online versions of $P|r_j| \sum_j w_j C_j$ and $P|r_j, pmtn| \sum_j w_j C_j$.

In online optimization we are dealing with limitations on information, in contrast with the limitations on computational power in classic approximation algorithm design. The standard measure of quality of online algorithms is the so-called competitive ratio. Analogous to the approximation guarantee of an algorithm, the competitive ratio is defined to be the worst-case ratio, over all instances, of the cost output by the online algorithm to the optimal offline cost. We shall also say that an online algorithm is c-competitive if, for any instance, the cost output by the online algorithm is at most c times the optimal offline cost. We may alternatively define the competitive ratio of an online algorithm as the infimum of all values c, for which the algorithm is c-competitive. In some situations randomization is a powerful tool to obtain algorithms with better performance ratios. The competitive ratio of a randomized online algorithm is defined as above, except that we replace "the cost output by the online algorithm" by "the *expected* cost output by the online algorithm."

1.1 Previous Work

There has been an enormous amount of work on parallel machine scheduling. We do not intend to do a complete review of results in the area and restrict our attention to some of the most relevant literature on online scheduling problems directly related to the matter of this paper.

To the best of our knowledge, the first deterministic online algorithm for $P|r_j| \sum_j w_j C_j$ was given by Hall et al. [13]. They design a $(4 + \varepsilon)$-competitive algorithm. Prior to this paper, the best-known deterministic algorithms for both $P|r_j| \sum_j w_j C_j$ and $P|r_j, pmtn| \sum_j w_j C_j$ were recently given by Megow and Schulz [17] and are 3.28 and 2-competitive, respectively. They also show that the former algorithm has a competitive ratio between 2.78 and 3.28 while the latter analysis is tight.

Considering randomized algorithms, a $(2.89 + \varepsilon)$-competitive algorithm for $P|r_j| \sum w_j C_j$ was obtained by Chakrabarti et al. [4]. Schulz and Skutella [21] give randomized strategies that are 2-competitive for both $P|r_j, pmtn| \sum_j w_j C_j$ and $P|r_j| \sum_j w_j C_j$. Related results have been obtained by Chekuri et al. [5] and Phillips et al. [18], when the objective is to minimize the average completion time of the schedule. In a more restricted setting, Chou et al. [6] consider the online $P|r_j| \sum_j w_j C_j$

with lower and upper bounds on jobs' weights and processing times; the authors prove that the online *weighted shortest processing time* heuristic is asymptotically optimal. They even extend this to the problem $Q|r_j|\sum_j w_j C_j$.

We also mention some single machine scheduling results, as our work essentially extends these analyses to the parallel machine case. Using the idea of α-points and mean-busy-time relaxations, Goemans et al. [11] designed a deterministic 2.4143-competitive and a randomized 1.6853-competitive algorithm for the online $1|r_j|\sum_j w_j C_j$. A similar approach was taken by Schulz and Skutella [22] to give a randomized $\frac{4}{3}$-competitive algorithm for $1|r_j, pmtn|\sum_j w_j C_j$; Sitters [24] gave a 1.56-competitive deterministic algorithm for the same problem. On the other hand, Anderson and Potts [3] provide a best possible deterministic online algorithm for $1|r_j|\sum_j w_j C_j$ which has a competitive ratio of 2. Additionally, Savelsbergh et al. [19] perform an extensive computational study of a number of heuristics and approximation algorithms for the offline single machine problem $1|r_j|\sum_j w_j C_j$. They conclude that LP-based approximation algorithms for this problem perform very well.

A crucial tool for our paper is list scheduling, where the order in which jobs are processed (on one or many machines) is determined by their priority order in a given list. Perhaps the most well-known list-scheduling rule is Smith's ratio rule [26], which schedules jobs on a single machine in non-decreasing order of the ratios p_j/w_j and is optimal for $1||\sum w_j C_j$. The incorporation of release dates into scheduling problems suggests the potential of preemptive list scheduling. For example, in an environment where preemption is allowed, if the highest priority job in a list is unavailable due to a release date constraint, it makes sense to process a lower priority job until the higher priority job is available. Therefore, a preemptive list schedule, where the highest priority *available* jobs are processed, can be appropriate in many cases. We utilize both non-preemptive and preemptive list schedules in this paper. In particular, the idea of creating a parallel machine list schedule by considering a single-machine relaxation is central to our analysis. This idea is implicitly contained in the work of Eastman et al. [8], and lately has been used extensively as an algorithmic tool [5,6,13,21].

We next discuss the development of the α-point concept, which we utilize to create preemptive and non-preemptive list schedules for assigning jobs to parallel identical machines. The α-point of a job is the first point in time that a fraction $\alpha \in (0, 1]$ of the job's processing requirement has been completed. The idea to list-schedule in the order of jobs' α-points has proven to be a powerful tool. The first heuristic application of α-points is found in de Sousa [7]. The idea of applying α-points to construct provably good schedules (in the offline setting) was introduced by Phillips et al. [18] and by Hall et al. [13]. Subsequently, the idea to choose α randomly was investigated by Goemans [10] and Chekuri et al. [5]. Additionally, the idea to randomly choose *different* values of α for each job was successfully applied in Goemans et al. [11]. A thorough survey of α-points in single machine scheduling is found in Skutella [25].

Another concept relevant to our analysis is the *mean busy time* of a job, which is defined as the average point in time that a job is processed. This idea was first introduced by Goemans [10] (see also [11]). Jobs' mean busy times were utilized extensively in Chou et al. [6]. The application of mean busy times also appears in Schulz [20], which extends the analysis in this paper to *stochastic* parallel machine

scheduling (in both offline and online settings). Kovács and Beck [16] also apply the mean-busy-time concept.

Let us now discuss some lower bounds on the competitive ratios for certain problems. Hoogeveen and Vestjens [15] showed that there is no deterministic algorithm with competitive ratio strictly better than 2 for $1|r_j|\sum_j w_j C_j$. On the other hand Stougie and Vestjens [27] showed that $\frac{e}{e-1}$ is a lower bound on the competitive ratio of online randomized algorithms for the same problem. In the parallel machine case, Vestjens [28] proved that any deterministic algorithm for $P|r_j|\sum_j w_j C_j$ (respectively, $P|r_j, pmtn|\sum_j w_j C_j$) has a competitive ratio of at least 1.309 (resp. $\frac{22}{21}$). Seiden [23] proved that any randomized algorithm for $P|r_j|\sum_j w_j C_j$ has a competitive ratio of at least 1.157. To the best of our knowledge, there are no specific lower bounds for randomized algorithms for $P|r_j, pmtn|\sum_j w_j C_j$.

It is worth mentioning that many of the results cited above have their origin in the analysis of offline algorithms. In some cases offline algorithms can be modified to function online and several offline approximation ratios can be transferred to online competitive ratios. Finally, we remark that all the problems considered in this paper admit polynomial time approximation schemes in the offline setting; see the paper by Afrati et al. [1].

1.2 Overview and structure of the paper

We begin in Sect. 2 by giving notation and existing results that are utilized throughout the paper. Sections 3–5 present our main results, which are detailed in the following paragraphs. Finally, although this is essentially a theoretical paper, Sect. 6 provides a brief computational study of the algorithms presented, under randomly chosen instances.

In Sect. 3, we generalize the ideas given by Goemans et al. [11] for the single machine case to the parallel machine problem $P|r_j|\sum_j w_j C_j$. As in that paper, our algorithm simulates a preemptive single machine scheduling problem on a virtual machine that is m times faster (where m is the number of machines). As soon as a fraction α of a given job has been processed in the fast machine, the algorithm will put such a job in a FIFO queue and will schedule it in the parallel machines at the first point in time at which a machine is idle and all jobs with higher priority have been assigned. By choosing an appropriate value of α, namely $\alpha = (\sqrt{5}-1)/2$, we can prove that our algorithm is 2.618-competitive, improving upon the 3.28-competitive algorithm of Megow and Schulz [17]. Our algorithm is deterministic and works online.

As in Goemans et al. [11] we will show that the algorithm just described can be improved with the help of randomization. Basically, instead of taking a fixed value of α for all jobs, we can choose different α_j's for different jobs, and moreover, choose these values at random according to a given distribution. This is shown in Sect. 4, where we give a randomized ϱ_m-competitive online algorithm for $P|r_j|\sum_j w_j C_j$, where $\varrho_m < 2$ for all $m \geq 1$. Here, m denotes the number of machines and ϱ_m is obtained implicitly. Our result improves upon the 2-competitive randomized algorithm by Schulz and Skutella [21]. In contrast to their work, our algorithm has the desirable property of being a list-scheduling algorithm that uses only one step of randomization for each

Table 1 Best known competitive ratios for a variety of problems

Problem	Deterministic	Randomized
$1\|r_j\|\sum_j w_j C_j$	2 [3]	1.69 [11]
$1\|r_j, pmtn\|\sum_j w_j C_j$	1.56 [24]	1.33 [22]
$P\|r_j\|\sum_j w_j C_j$	3.28 [17]	2 [21]
$P\|r_j, pmtn\|\sum_j w_j C_j$	2 [17]	2 [21]

Table 2 Competitive ratios given in this paper

Problem	Deterministic	Randomized
$P\|r_j\|\sum_j w_j C_j$	2.62	$\varrho_m < 2,\ \forall m \geq 1$
$P\|r_j, pmtn\|\sum_j w_j C_j$	–	$2 - 1/m,\ m \geq 3;\ 1.5225,\ m = 2$

job (in [21], each job is randomly assigned a value of α *and* is randomly assigned to a machine). The algorithm we present can be seen as the parallel machine extension of the algorithm of Goemans et al. [11] for a single machine. Indeed, the competitive ratio that it achieves is 1.6853 for $m = 1$ (as in Goemans et al.); for $m = 2$, 3 and 4 it is 1.8382, 1.8915 and 1.9184, respectively (Table 1).

Following the algorithmic idea above, in Sect. 5 we present a randomized ρ_m-competitive online algorithm for $P|r_j, pmtn|\sum_j w_j C_j$, where $\rho_m = 2 - 1/m$ for $m \geq 3$ and $\rho_2 < 1.5225$. The reader may wish to compare our result with the current best algorithm to date: the deterministic algorithm by Megow and Schulz [17], which has a competitive ratio of 2 (and not better than 2) for any number of machines. Additionally, our algorithm can be simultaneously seen as an extension of Megow and Schulz's result and of Schulz and Skutella's [22] single machine algorithm. Indeed for a single machine, the competitive ratio of our algorithm is 4/3, as in [22], and approaches 2 as the number of machines grows to infinity (Table 2).

Finally, in Sect. 6, we present a brief computational study. Interestingly, these computational results suggest that, in practice, the schedules output by our algorithms will be much better than what the previous theoretical bounds predict.

2 Preliminaries

According to Phillips et al. [18] the α-point $t_j(\alpha)$, $0 < \alpha \leq 1$, of job j in a given schedule is defined as the first time an α-fraction of job j has been completed (i.e., the first time when αp_j units have been processed). The general idea of our subsequent algorithms is to schedule jobs on the m machines by list-scheduling the jobs in the order of their α-points on a virtual machine, which is "m-times faster". Additionally, these algorithms may use job-dependent α's to guide the schedule; in this latter case, we shall denote job j's alpha value as α_j. The concept of a single fast virtual machine was apparently first considered by Eastman et al. [8]. Recently, Chekuri et al. [5] considered a "preemptive one-machine relaxation" where jobs are list-scheduled on parallel machines in order of their completion times on a single virtual machine.

Another important ingredient in what follows is related to mean-busy-time relaxations of $1|r_j|\sum w_jC_j$. The *mean busy time* M_j of job j is defined as the average point in time at which job j is being processed; see Goemans [10] and Goemans et al. [11]. Letting $1_j(t) = 1$ if the machine is processing job j at time t and $1_j(t) = 0$ otherwise, the mean busy time M_j can be calculated as

$$M_j = \frac{1}{p_j}\int_0^\infty t1_j(t)dt.$$

Alternatively, it can be computed as $M_j = \int_0^1 t_j(\alpha)d\alpha$. Let $p(S) = \sum_{j\in S} p_j$, $w(S) = \sum_{j\in S} w_j$ and $r_{\min}(S) = \min_{j\in S}\{r_j\}$. Following Goemans et al. [11], for a scheduling instance $I = \{(p_i, r_i, w_i),\ i \in N\}$ we define $Z_R(I)$ to be the value of the mean-busy-time relaxation for $1|r_j, pmtn|\sum_j w_jC_j$:

$$Z_R(I) \triangleq \min \sum_{j\in N} w_jM_j$$
$$\text{subject to} \sum_{j\in S} p_jM_j \geq p(S)\left(r_{\min}(S) + \frac{1}{2}p(S)\right),\ S \subseteq N.$$

It was shown in [10] that $Z_R(I)$ can be obtained online by scheduling, at any point in time, the available job j with the highest ratio w_j/p_j (with p_j being the original processing time requirement). This schedule is called the *LP schedule*.

Now, for an instance $I = \{(p_i, r_i, w_i),\ i \in N\}$ of $P|r_j, pmtn|\sum w_jC_j$ with m parallel machines, let $Z^m(I)$ be the value of the optimal schedule. Consider the instance $I_m = \{(\frac{p_i}{m}, r_i, w_i),\ i \in N\}$ and let $Z^m_R(I) = Z_R(I_m)$, i.e., the value of the mean-busy-time relaxation on I_m (note that this is equivalent to the value of the mean-busy-time relaxation on instance I in a machine that is m times faster). Thus, $Z^m_R(I)$ can be evaluated as

$$Z^m_R(I) = \min \sum_{j\in N} w_jM_j$$
$$\text{subject to} \sum_{j\in S} p_jM_j \geq p(S)\left(r_{\min}(S) + \frac{1}{2m}p(S)\right),\quad S \subseteq N.$$

The following lemma provides a simple, yet powerful, lower bound for $P|r_j, pmtn|\sum w_jC_j$. It is a particular case of a bound obtained by Chou et al. [6] in a more general framework. It was also obtained by Schulz and Skutella [21], expressed in terms of an equivalent time-indexed relaxation.

Lemma 2.1 **([6,21])** *For any scheduling instance I, $Z^m_R(I) + \frac{1}{2}\sum_{j\in N} w_jp_j \leq Z^m(I)$.*

To finish this section let us review the concept of canonical decomposition as introduced by Goemans [9] and a useful formula to rewrite $\sum w_jM_j$ [10] (see also

[11]). For a set of jobs S, consider a single machine schedule that processes jobs in S as early as possible. This induces a partition of jobs in S into $S_1, \ldots, S_k$ such that the machine is busy exactly in the disjoint intervals $[r_{\min}(S_l), r_{\min}(S_l) + p(S_l)]$, for $l = 1, \ldots, k$. This partition is the *canonical decomposition* of S. Also, a set S is called canonical if its canonical decomposition consists of the single set S. Assume that $w_1/p_1 \geq \cdots \geq w_n/p_n \geq w_{n+1}/p_{n+1} = 0$ and let $[i] = \{1, \ldots, i\}$. Consider $S_1^i, \ldots, S_{k(i)}^i$, the canonical decomposition of $[i]$; then for any vector $M = (M_1, \ldots, M_n)$,

$$\sum_{j \in N} w_j M_j = \sum_{i=1}^{n} \left(\frac{w_i}{p_i} - \frac{w_{i+1}}{p_{i+1}} \right) \sum_{j \in [i]} p_j M_j = \sum_{i=1}^{n} \left(\frac{w_i}{p_i} - \frac{w_{i+1}}{p_{i+1}} \right) \sum_{l=1}^{k(i)} \sum_{j \in S_l^i} p_j M_j. \tag{1}$$

3 A deterministic online algorithm for $P|r_j|\sum w_j C_j$

Consider the following online algorithm *Non-preemptive α scheduling* (NAS), where each job j is assigned a deterministic value of α_j.

Algorithm NAS:

INPUT: A scheduling instance $I = \{(p_i, r_i, w_i),\ i \in N\}$ that is revealed online, and a vector $\boldsymbol{\alpha} = \{\alpha_1, \ldots, \alpha_n\}$.

(1) Construct the preemptive LP-schedule on a single virtual machine m-times faster ($I \mapsto I_m$).

(2) At job j's α_j-point $t_j(\alpha_j)$ in the virtual machine, it enters into a FIFO queue for the m machines (job j is then scheduled the first time a machine is available after all preceding jobs in the queue have started).

From now on, whenever we refer to the LP-schedule of instance I, we mean the LP-schedule in a machine that is m times faster (or the LP-schedule of I_m). Consider job j and let $\eta_k(\alpha_j)$ denote the fraction of job k that has been completed in the LP-schedule by time $t_j(\alpha_j)$. Letting $C_j^{\boldsymbol{\alpha}}$ denote the completion time of job j in algorithm NAS when the vector $\boldsymbol{\alpha} = \{\alpha_1, \ldots, \alpha_n\}$ is applied, we can show the following bound. Bounds of similar flavor have been frequently used in the scheduling literature (e.g. [11,13,18]).

Lemma 3.1

$$C_j^{\boldsymbol{\alpha}} \leq t_j(\alpha_j) + \sum_{k:\alpha_k \leq \eta_k(\alpha_j)} \frac{p_k}{m} + \left(1 - \frac{1}{m}\right) p_j.$$

Proof The completion time of job j equals the time to enter the queue for the parallel machines plus the waiting time in queue plus the processing time of job j.

The time to enter the queue is $t_j(\alpha_j)$, which is the α_j-point of job j in the single virtual machine that is m-times faster.

The wait time in the queue can be bounded as follows: Consider all jobs that entered the queue before job j, i.e., jobs belonging to the set $\{k \neq j : \alpha_k \leq \eta_k(\alpha_j)\}$ (which

are all available for processing at time $t_j(\alpha_j)$ or earlier). Then the total work that needs to be processed before job j in the m machines is at most $\sum_{k\neq j:\alpha_k\leq\eta_k(\alpha_j)} p_k$. Thus the first time that a machine will free up is at most

$$t_j(\alpha_j) + \frac{\sum_{k\neq j:\alpha_k\leq\eta_k(\alpha_j)} p_k}{m} = t_j(\alpha_j) - \frac{p_j}{m} + \sum_{k:\alpha_k\leq\eta_k(\alpha_j)} \frac{p_k}{m},$$

which is obtained by averaging the processing times of all jobs before j. Adding up the previous term with the processing time p_j gives the result. □

Our deterministic algorithm will perform best by taking a fixed value of α for all jobs: $\alpha_j = \alpha,\ \forall j$. The following theorem is the main result of this section. Its proof is an extension of the proof of Theorem 3.3 in [11] to the parallel machine case.

Theorem 3.2 *Algorithm NAS is* $\max\{1+\frac{1}{\alpha}, 2+\alpha\}$*-competitive. In particular, for* $\alpha = \frac{\sqrt{5}-1}{2}$*, the schedule is* $\left(\frac{3+\sqrt{5}}{2}\right)$*-competitive* $\left(\frac{3+\sqrt{5}}{2} < 2.6181\right)$.

Proof Consider a canonical set $S = \{1,\ldots,l\}$ for the fast single machine. Fix a job $j \in S$ and let $\eta_k = \eta_k(\alpha)$ represent the fraction of job k processed before $t_j(\alpha)$. By reordering the elements in S such that $t_1(\alpha) \leq \cdots \leq t_l(\alpha)$, we have that

$$t_j(\alpha) - r_{\min}(S) = \sum_{k\in S} \eta_k \frac{p_k}{m} \leq \sum_{k=j}^{l} \alpha \frac{p_k}{m} + \sum_{k=1}^{j-1} \frac{p_k}{m} = \frac{\alpha}{m} p(S) + \frac{(1-\alpha)}{m} \sum_{k=1}^{j-1} p_k. \tag{2}$$

Let C_j^α be the completion time of job j output by algorithm NAS. Define R to be the set of jobs k such that $t_k(\alpha) < r_{\min}(S)$; note that $R \bigcap S = \emptyset$ and $R \bigcup \{1,\ldots,j\} = \{k : \alpha \leq \eta_k\}$. Thus, combining Lemma 3.1 with Eq. (2) and then noting that $\alpha\frac{p(R)}{m} \leq r_{\min}(S)$, we get

$$\begin{aligned} C_j^\alpha &\leq r_{\min}(S) + \frac{\alpha}{m} p(S) + \frac{(1-\alpha)}{m} \sum_{k=1}^{j-1} p_k + \frac{1}{m} p(R) + \frac{1}{m} \sum_{k=1}^{j-1} p_k + p_j \\ &\leq \left(1+\frac{1}{\alpha}\right) r_{\min}(S) + \frac{\alpha}{m} p(S) + \frac{(2-\alpha)}{m} \sum_{k=1}^{j-1} p_k + p_j. \end{aligned}$$

Multiplying by p_j and summing over S we get

$$\sum_{j\in S} p_j C_j^\alpha \leq \left(1+\frac{1}{\alpha}\right) r_{\min}(S) p(S) + \frac{\alpha}{m} p(S)^2 + \frac{(2-\alpha)}{m} \sum_{j\in S} \sum_{k=1}^{j-1} p_j p_k + \sum_{j\in S} p_j^2.$$

Using the identity $\sum_{j\in S}\sum_{k=1}^{j-1} p_j p_k = \frac{1}{2}p(S)^2 - \frac{1}{2}\sum_{j\in S} p_j^2$ we obtain that for any canonical set S,

$$\begin{aligned}\sum_{j\in S} p_j C_j^{\alpha} &\le \left(1+\frac{1}{\alpha}\right) r_{\min}(S)p(S) + (2+\alpha)\frac{p(S)^2}{2m} + \sum_{j\in S} p_j^2 \\ &\le \max\left\{1+\frac{1}{\alpha}, 2+\alpha\right\}\left(p(S)\left(r_{\min}(S)+\frac{p(S)}{2m}\right)+\frac{1}{2}\sum_{j\in S} p_j^2\right).\end{aligned}$$

Assume now that the jobs are ordered such that $w_1/p_1 \ge \cdots \ge w_n/p_n \ge w_{n+1}/p_{n+1} = 0$. Let us now bound the overall cost of the schedule using Eq. (1) applied to instance I_m and the feasibility of the vector $M = (M_1, \ldots, M_n)$ for $Z_R^m(I)$:

$$\begin{aligned}\sum_{j\in N} w_j C_j^{\alpha} &= \sum_{i=1}^{n}\left(\frac{w_i}{\frac{p_i}{m}} - \frac{w_{i+1}}{\frac{p_{i+1}}{m}}\right)\sum_{l=1}^{k(i)}\sum_{j\in S_l^i}\frac{p_j}{m}C_j^{\alpha} \\ &\le \max\left\{1+\frac{1}{\alpha}, 2+\alpha\right\}\sum_{i=1}^{n}\left(\frac{w_i}{\frac{p_i}{m}} - \frac{w_{i+1}}{\frac{p_{i+1}}{m}}\right) \\ &\quad\times\sum_{l=1}^{k(i)}\left(\frac{p(S_l^i)}{m}\left(r_{\min}(S_l^i)+\frac{p(S_l^i)}{2m}\right)+\frac{1}{2m}\sum_{j\in S_l^i} p_j^2\right) \\ &\le \max\left\{1+\frac{1}{\alpha}, 2+\alpha\right\}\sum_{i=1}^{n}\left(\frac{w_i}{\frac{p_i}{m}} - \frac{w_{i+1}}{\frac{p_{i+1}}{m}}\right) \\ &\quad\times\sum_{l=1}^{k(i)}\left(\sum_{j\in S_l^i}\frac{p_j}{m}M_j+\frac{1}{2m}\sum_{j\in S_l^i} p_j^2\right) \\ &= \max\left\{1+\frac{1}{\alpha}, 2+\alpha\right\}\sum_{i=1}^{n}\left(\frac{w_i}{\frac{p_i}{m}} - \frac{w_{i+1}}{\frac{p_{i+1}}{m}}\right)\sum_{l=1}^{k(i)}\sum_{j\in S_l^i}\frac{p_j}{m}\left(M_j+\frac{p_j}{2}\right).\end{aligned}$$

Here, M_j denotes the mean busy time of job j in the LP-schedule. Applying Eq. (1) again it follows that the previous quantity equals

$$\begin{aligned}&\max\left\{1+\frac{1}{\alpha}, 2+\alpha\right\}\sum_{j\in N} w_j\left(M_j+\frac{p_j}{2}\right) \\ &\quad= \max\left\{1+\frac{1}{\alpha}, 2+\alpha\right\}\left(Z_R^m(I)+\frac{1}{2}\sum_{j\in N} w_j p_j\right),\end{aligned}$$

and by Lemma 2.1 it follows that $\sum_{j\in N} w_j C_j^{\alpha} \leq \max\{1+\frac{1}{\alpha}, 2+\alpha\} Z^m(I)$. Finally, we recall that $Z^m(I)$, the optimal value of $P|r_j, pmtn|\sum_j w_j C_j$, is a lower bound on the optimal offline cost of $P|r_j|\sum_j w_j C_j$. □

As in the single machine case, there are instances for which algorithm NAS gives a schedule with cost as much as twice the LP lower bound; see, for example, [11]. However, we do not know whether our analysis is tight.

4 A randomized online algorithm for $P|r_j|\sum_j w_j C_j$

Consider the following algorithm, which we denote as *Non-preemptive α scheduling randomized* (NASR).

Algorithm NASR:

INPUT: A scheduling instance $I = \{(p_i, r_i, w_i),\ i \in N\}$ that is revealed online, and a distribution f.

(1) Construct the preemptive LP-schedule on a single virtual machine m-times faster ($I \mapsto I_m$).
(2) Each α_j is taken identically and independently from distribution $f(\alpha)$.
(3) At job j's α_j-point $t_j(\alpha_j)$, it enters into a FIFO queue for the m machines (job j is then scheduled the first time a machine is available after all preceding jobs in the queue have started).

We start by proving that the uniform distribution already gives 2-competitiveness; this matches Schulz and Skutella's [21] bound with a list-scheduling algorithm. The proof is very similar to Theorem 3.4 in [11].

Theorem 4.1 *NASR is 2-competitive when $f(\alpha)$ is the uniform distribution on* (0, 1].

Proof Let $C_j^{\boldsymbol{\alpha}}$ be the completion time of job j in the schedule given by algorithm NASR. We apply Lemma 3.1 and first consider a conditional expectation, holding α_j constant:

$$\mathrm{E}\left[C_j^{\boldsymbol{\alpha}}|\alpha_j\right] \leq t_j(\alpha_j) + \sum_{k\neq j} \frac{p_k}{m} \int_0^{\eta_k(\alpha_j)} d\alpha_k + p_j = t_j(\alpha_j)$$
$$+ \sum_{k\neq j} \frac{p_k}{m} \eta_k(\alpha_j) + p_j \leq 2\left(t_j(\alpha_j) + \frac{p_j}{2}\right).$$

This implies that

$$\mathrm{E}\left[C_j^{\boldsymbol{\alpha}}\right] \leq \int_0^1 2\left(t_j(\alpha_j) + \frac{1}{2}p_j\right) d\alpha_j = 2\left(M_j + \frac{1}{2}p_j\right),$$

where M_j denotes the mean busy time of job j in the LP-schedule. Multiplying by w_j and summing over j we get

$$\mathrm{E}\left[\sum_{j\in N} w_j C_j^{\boldsymbol{\alpha}}\right] \le 2\left(Z_R(I_m) + \frac{1}{2}\sum_{i\in N} w_i p_i\right) \le 2\cdot Z^m(I),$$

which proves the result. □

We now turn to deriving improved bounds which will depend on the number of machines. We show that by taking the α_j from an appropriate distribution we can improve on 2-competitiveness. Let us start by giving a refinement of Lemma 3.1.

Lemma 4.2

$$C_j^{\boldsymbol{\alpha}} \le t_j(\alpha_j) + \sum_{\substack{k:\alpha_k\le\eta_k(\alpha_j)\\ k\neq j}} \left(1 + \frac{\alpha_k}{m} - \frac{\eta_k(\alpha_j)}{m}\right)\frac{p_k}{m} + p_j.$$

Proof As in Lemma 3.1, the completion time of job j is equal to the time to enter the queue for the parallel machines plus the wait-time in queue plus the processing time of job j. The only difference in the bound we are attempting to prove here lies in the in-queue waiting time. This can be bounded as follows:

Consider the set K of jobs that entered the queue before job j; i.e., $K = \{k : \alpha_k \le \eta_k(\alpha_j), k \neq j\}$. If at time $t > t_k(\alpha_k)$ the fast machine is processing job k, then at least one of the parallel machines is busy at time t (maybe processing another job). Indeed, even if job k is interrupted by a job, say l, the fast machine will only go back to processing job k after l is completed; thus l will enter the queue before time t.

Thus, at time $t_j(\alpha_j)$, the parallel machines have together processed $\sum_{k\in K}(\eta_k(\alpha_j) - \alpha_k)\frac{p_k}{m}$ units of work. Now, the total processing requirement entered into the queue before job j is $\sum_{k\in K} p_k$. Since we have just argued that by time $t_j(\alpha_j)$, the m machines have processed $\sum_{k\in K}(\eta_k(\alpha_j) - \alpha_k)\frac{p_k}{m}$, the remaining processing requirement in the system at time $t_j(\alpha_j)$ is

$$\sum_{k\in K} p_k - \sum_{k\in K}(\eta_k(\alpha_j) - \alpha_k)\frac{p_k}{m} = \sum_{k\in K}\left(1 + \frac{\alpha_k}{m} - \frac{\eta_k(\alpha_j)}{m}\right)p_k.$$

Using standard averaging arguments, the first time a machine will empty up to process job j is at most $\sum_{k\in K}\left(1 + \frac{\alpha_k}{m} - \frac{\eta_k(\alpha_j)}{m}\right)\frac{p_k}{m}$. □

Recall that N is defined as the set of all jobs. For a given job j, we partition $N\backslash\{j\}$ into N_1 and N_2. N_2 is the set of all jobs that are processed between the start and completion of job j on the fast virtual machine and N_1 consists of any remaining jobs.

For any $k \in N_2$, we let μ_k denote the fraction of job j that, in the LP schedule of I_m, is processed before the start of job k. This implies $\forall k \in N_2$

$$\eta_k(\alpha_j) = \begin{cases} 0, & \alpha_j \leq \mu_k \\ 1, & \alpha_j > \mu_k. \end{cases}$$

Letting $t_j(0^+)$ denote the start time of job j on the fast virtual machine, we may then write

$$t_j(\alpha_j) = t_j(0^+) + \sum_{\substack{k \in N_2 \\ \alpha_j > \mu_k}} \frac{p_k}{m} + \alpha_j \frac{p_j}{m}.$$

Recalling that in the LP-schedule $M_j = \int_0^1 t_j(\alpha) d\alpha$ we have that

$$M_j = t_j(0^+) + \sum_{k \in N_2} (1 - \mu_k) \frac{p_k}{m} + \frac{1}{2} \frac{p_j}{m}. \tag{3}$$

We can now rewrite Lemma 4.2 as

$$\begin{aligned} C_j^{\boldsymbol{\alpha}} \leq t_j(0^+) + & \sum_{\substack{k \in N_1 \\ \alpha_k \leq \eta_k(\alpha_j)}} \left(1 + \frac{\alpha_k}{m} - \frac{\eta_k(\alpha_j)}{m}\right) \frac{p_k}{m} \\ & + \sum_{\substack{k \in N_2 \\ \alpha_j > \mu_k}} \left(2 - \frac{1 - \alpha_k}{m}\right) \frac{p_k}{m} + \left(1 + \frac{\alpha_j}{m}\right) p_j. \end{aligned} \tag{4}$$

This bound will prove useful in the proof of the main result in this section.

For any $m \geq 1$, consider the following equation, which extends the equation in Theorem 3.9 in [11] to an arbitrary number of machines:

$$\begin{aligned} & \ln\left(1 + \frac{1}{m} - \frac{\gamma}{m}\right) + \frac{\gamma}{m} \\ & = \frac{e^{(-\gamma/m)} \left(1 + \frac{1}{m} - \frac{\gamma}{m} - e^{(-\gamma/m)}\right) \left(m e^{(\gamma/m)} - \gamma e^{(\gamma/m)} + \frac{1}{m} e^{(\gamma/m)} + 1 - m\right)}{1 + \frac{1}{m} - \frac{\gamma}{m}}. \end{aligned} \tag{5}$$

For any finite value of m, it can be shown that Eq. (5) has an unique solution $\gamma \in (0, 1)$ (see Appendix 6.4). We set

$$\frac{\delta_m}{m} = \ln\left(1 + \frac{1}{m} - \frac{\gamma}{m}\right) + \frac{\gamma}{m},$$

for the unique value of γ that satisfies Eq. (5). It can also be shown that $\delta_m \in (0, 1)$ for any finite m (see Appendix 6.4). With this, we can consider the following distribution:

$$f(\alpha) = \begin{cases} c_m e^{(\alpha/m)}, & 0 \le \alpha \le \delta_m \\ 0, & o.w. \end{cases}$$

where $c_m = \left(m\left(e^{\delta_m/m} - 1\right)\right)^{-1}$. The main result is then the following.

Theorem 4.3 *With $f(\alpha)$ as above, NASR is $(1 + c_m)$-competitive.*

To prove this theorem, we first need a series of technical lemmas.

Lemma 4.4 $\int_0^{\eta} f(\alpha)\left(1 + \frac{\alpha}{m} - \frac{\eta}{m}\right) d\alpha \le c_m \eta, \ \forall \eta \in [0, 1].$

Proof For $\eta \in [0, \delta_m]$,

$$\int_0^{\eta} f(\alpha)\left(1 + \frac{\alpha}{m} - \frac{\eta}{m}\right) d\alpha = c_m\left[(m - \eta)\left(e^{(\eta/m)} - 1\right) + \frac{1}{m}\left(m\eta e^{(\eta/m)} - m^2 e^{(\eta/m)} + m^2\right)\right] = c_m \eta.$$

For $\eta \in (\delta_m, 1]$,

$$\int_0^{\eta} f(\alpha)\left(1 + \frac{\alpha}{m} - \frac{\eta}{m}\right) d\alpha < \int_0^{\delta_m} f(\alpha)\left(1 + \frac{\alpha}{m} - \frac{\delta_m}{m}\right) d\alpha = c_m \delta_m < c_m \eta.$$

□

Lemma 4.5 $\left(2 - \frac{(1 - \mathrm{E}[\alpha])}{m}\right) \int_{\mu}^{1} f(\alpha) d\alpha \le (c_m + 1)(1 - \mu), \ \forall \mu \in [0, 1].$

Proof We first find the expected value of α; $\mathrm{E}[\alpha] = (mc_m + 1)\delta_m - m$. Therefore,

$$2 - \frac{(1 - \mathrm{E}[\alpha])}{m} = 1 - \frac{1}{m} + (mc_m + 1)\frac{\delta_m}{m}.$$

We now turn to the other component. For $\mu \in [0, \delta_m]$,

$$\int_{\mu}^{1} f(\alpha) d\alpha = c_m m\left(e^{(\delta_m/m)} - e^{(\mu/m)}\right).$$

Thus,

$$\left(2-\frac{(1-\mathrm{E}[\alpha])}{m}\right)\int_{\mu}^{1} f(\alpha)d\alpha = \left(1-\frac{1}{m}+(mc_m+1)\frac{\delta_m}{m}\right)c_m m\left(e^{(\delta_m/m)}-e^{(\mu/m)}\right).$$

We now use Eq. (5) and calculate the portion

$$\left(1-\frac{1}{m}+(mc_m+1)\frac{\delta_m}{m}\right)c_m. \tag{6}$$

Define $\Psi = 1+\frac{1}{m}-\frac{\gamma}{m}-e^{(-\gamma/m)}$ and $\Upsilon = me^{(\gamma/m)}-\gamma e^{(\gamma/m)}+\frac{1}{m}e^{(\gamma/m)}+1-m$, so that the RHS of Eq. (5) can be written as $\frac{e^{(-\gamma/m)}\Psi\Upsilon}{(1+\frac{1}{m}-\frac{\gamma}{m})}$. Now, using the fact that $e^{(\delta_m/m)} = e^{(\gamma/m)}(1+\frac{1}{m}-\frac{\gamma}{m})$, we have that $c_m = \frac{e^{-(\gamma/m)}}{m\Psi}$.

Substituting this expression for c_m in Expression (6) and replacing $\frac{\delta_m}{m}$ with the RHS of Eq. (5), we have that

$$\begin{aligned}
\left(1-\frac{1}{m}+(mc_m+1)\frac{\delta_m}{m}\right)c_m &= \left(1-\frac{1}{m}+\left(\frac{e^{-(\gamma/m)}}{\Psi}+1\right)\frac{\delta_m}{m}\right)\frac{e^{-(\gamma/m)}}{m\Psi}\\
&= \left(1-\frac{1}{m}+\left(\frac{e^{-(\gamma/m)}}{\Psi}+1\right)\frac{e^{(-\gamma/m)}\Psi\Upsilon}{(1+\frac{1}{m}-\frac{\gamma}{m})}\right)\frac{e^{-(\gamma/m)}}{m\Psi}\\
&= \left(1-\frac{1}{m}+\left(e^{-(\gamma/m)}+\Psi\right)\frac{e^{(-\gamma/m)}\Upsilon}{(1+\frac{1}{m}-\frac{\gamma}{m})}\right)\frac{e^{-(\gamma/m)}}{m\Psi}\\
&= \left(1-\frac{1}{m}+e^{(-\gamma/m)}\Upsilon\right)\frac{e^{-(\gamma/m)}}{m\Psi}\\
&= \left(e^{-(\gamma/m)}+m\Psi\right)\frac{e^{-(\gamma/m)}}{m\Psi}\\
&= (c_m+1)e^{-(\gamma/m)}.
\end{aligned}$$

Consequently,

$$\begin{aligned}
\left(2-\frac{(1-\mathrm{E}[\alpha])}{m}\right)\int_{\mu}^{1} f(\alpha)d\alpha &= (c_m+1)e^{-(\gamma/m)}m\left(e^{(\delta_m/m)}-e^{(\mu/m)}\right)\\
&= (c_m+1)m\left(1+\frac{1}{m}-\frac{\gamma}{m}-e^{((\mu-\gamma)/m)}\right)\\
&\le (c_m+1)m\left(1+\frac{1}{m}-\frac{\gamma}{m}-1-\frac{(\mu-\gamma)}{m}\right)\\
&= (c_m+1)(1-\mu).
\end{aligned}$$

□

Lemma 4.6 $1 + \dfrac{\mathrm{E}[\alpha]}{m} \le (c_m + 1)\left(\dfrac{m+1}{2m}\right)$.

Proof A simple calculation yields

$$1 + \frac{\mathrm{E}[\alpha]}{m} = \frac{\delta_m e^{(\delta_m/m)}}{m\left(e^{(\delta_m/m)} - 1\right)} = \frac{2m\delta_m e^{(\delta_m/m)}}{2m^2\left(e^{(\delta_m/m)} - 1\right)}. \tag{7}$$

Turning to the RHS of the lemma, we see that

$$c_m + 1 = \frac{m e^{(\delta_m/m)} - m + 1}{m\left(e^{(\delta_m/m)} - 1\right)},$$

and consequently

$$(c_m + 1)\left(\frac{m+1}{2m}\right) = \frac{m^2 e^{(\delta_m/m)} - m^2 + m e^{(\delta_m/m)} + 1}{2m^2\left(e^{(\delta_m/m)} - 1\right)}. \tag{8}$$

Examining the numerators of Eqs. (7) and (8) (and noting that the denominators are positive), it is sufficient to prove $m^2 e^{(\delta_m/m)} - m^2 + m e^{(\delta_m/m)} + 1 \ge 2m\delta_m e^{(\delta_m/m)}$; we demonstrate this as follows:

$$\begin{aligned}
&m^2 e^{(\delta_m/m)} - m^2 + m e^{(\delta_m/m)} + 1 - 2m\delta_m e^{(\delta_m/m)} \\
&= \left(m^2 + m - 2m\delta_m\right) e^{(\delta_m/m)} - m^2 + 1 \\
&\ge \left(m^2 + m - 2m\delta_m\right)\left(1 + \frac{\delta_m}{m}\right) - m^2 + 1 \\
&= m(1 - \delta_m) + \left(\delta_m - \delta_m^2\right) + \left(1 - \delta_m^2\right) \ge 0.
\end{aligned}$$

□

Now we prove Theorem 4.3.

Proof of Theorem 4.3 First, we fix α_j and use the shorthand $\eta_k = \eta_k(\alpha_j)$. Applying Eq. (4) and Lemma 4.4 we obtain

$$\begin{aligned}
\mathrm{E}\left[C_j^{\boldsymbol{\alpha}} \mid \alpha_j\right] &\le t_j(0^+) + \sum_{k \in N_1} \int_0^{\eta_k} f(\alpha_k)\left(1 + \frac{\alpha_k}{m} - \frac{\eta_k}{m}\right)\frac{p_k}{m} d\alpha_k \\
&\quad + \sum_{\substack{k \in N_2 \\ \alpha_j > \tilde{\mu}_k}} \left(2 - \frac{1 - \mathrm{E}[\alpha_k]}{m}\right)\frac{p_k}{m} + \left(1 + \frac{\alpha_j}{m}\right) p_j \\
&\le t_j(0^+) + c_m \sum_{k \in N_1} \eta_k \frac{p_k}{m} + \left(2 - \frac{1 - \mathrm{E}[\alpha]}{m}\right) \sum_{\substack{k \in N_2 \\ \alpha_j > \tilde{\mu}_k}} \frac{p_k}{m} + \left(1 + \frac{\alpha_j}{m}\right) p_j
\end{aligned}$$

$$\leq (c_m+1)t_j(0^+) + \left(2 - \frac{1-\mathrm{E}[\alpha]}{m}\right) \sum_{\substack{k\in N_2 \\ \alpha_j > \mu_k}} \frac{p_k}{m} + \left(1+\frac{\alpha_j}{m}\right) p_j.$$

The last inequality follows from the fact that $t_j(0^+) \geq \sum_{k\in N_1} \eta_k \frac{p_k}{m}$; this follows because, for every job $k \in N_1$, we have $\eta_k(\alpha_j) = \eta_k(0^+)$. In particular, if job k starts before job j and is not preempted by job j, then $\eta_k(\alpha_j) = 1 = \eta_k(0^+)$, and if job k is preempted by job j, then job j finishes before job k restarts, so we have $\eta_k(0^+) = \eta_k(\alpha_j) = \eta_k(1)$. We now integrate over α_j and apply Lemmas 4.5 and 4.6 to find a bound on the unconditional expectation:

$$\begin{aligned}
\mathrm{E}[C_j^{\boldsymbol{\alpha}}] &\leq (c_m+1)t_j(0^+) + \left(2-\frac{1-\mathrm{E}[\alpha]}{m}\right) \sum_{k\in N_2} \int_{\mu_k}^{1} f(\alpha_j)\frac{p_k}{m} d\alpha_j + \left(1+\frac{\mathrm{E}[\alpha_j]}{m}\right) p_j \\
&\leq (c_m+1)t_j(0^+) + (c_m+1) \sum_{k\in N_2} (1-\mu_k)\frac{p_k}{m} + \left(1+\frac{\mathrm{E}[\alpha_j]}{m}\right) p_j \\
&\leq (c_m+1)t_j(0^+) + (c_m+1) \sum_{k\in N_2} (1-\mu_k)\frac{p_k}{m} + (c_m+1)\left(\frac{m+1}{2m}\right) p_j \\
&= (c_m+1)\left(t_j(0^+) + \sum_{k\in N_2} (1-\mu_k)\frac{p_k}{m} + \frac{p_j}{2m} + \frac{p_j}{2}\right) = (c_m+1)\left(M_j + \frac{p_j}{2}\right),
\end{aligned}$$

where the last equality follows from Eq. (3). Multiplying by w_j and summing over j gives

$$\begin{aligned}
\mathrm{E}\left[\sum_{j\in N} w_j C_j^{\boldsymbol{\alpha}}\right] &\leq (c_m+1)\left(\sum_{j\in N} w_j M_j + \frac{1}{2}\sum_{j\in N} w_j p_j\right) \\
&= (c_m+1)\left(Z_R^m(I) + \frac{1}{2}\sum_{j\in N} w_j p_j\right) \\
&\leq (c_m+1) Z^m(I),
\end{aligned}$$

which proves the result. □

The proof of Theorem 4.3 shows that the cost of the non-preemptive algorithm NASR is at most $(1 + c_m)Z^m(I)$, where $Z^m(I)$ is the offline optimal value of $P|r_j, pmtn|\sum w_j C_j$. Therefore, we have the following corollary.

Corollary 4.7 *Algorithm NASR (with $f(\alpha)$ as in Theorem 4.3) returns a solution whose cost is within a factor $(1 + c_m)$ of the optimal offline preemptive schedule.*

The class of distributions we applied is optimal for our analysis. Essentially, Eq. (5) is a sufficient optimality condition for our distributions and analysis technique.

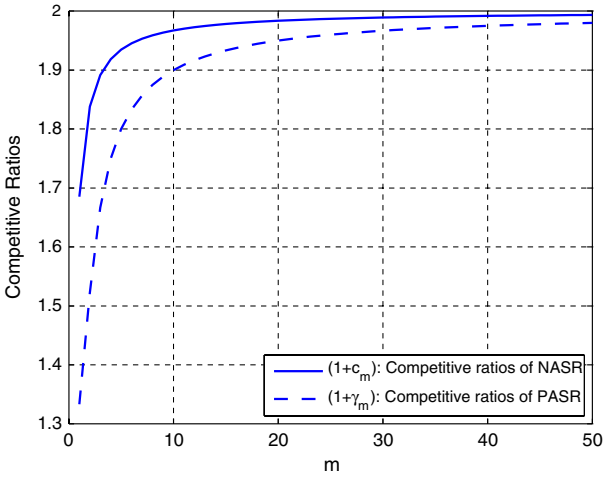

Fig. 1 The competitive ratios $(1 + c_m)$ and $(1 + \gamma_m)$ are plotted in the top (*solid line*) and bottom (*dashed line*) curves, respectively, as a function of m

We can also prove that $c_m < 1$ for any finite $m \geq 1$ and $\lim_{m\to\infty} c_m = 1$ (see Appendix 6.4). Not surprisingly then, as m grows, f uniformly approaches the uniform distribution on [0, 1]. Let us finish by plotting $1 + c_m$ as a function of the number of machines, as depicted in Fig. 1.

5 A randomized online algorithm for $P|r_j, pmtn|\sum_j w_j C_j$

We now consider the simpler preemptive case. Consider the following algorithm *Preemptive α scheduling randomized* (PASR).

Algorithm PASR:

INPUT: A scheduling instance $I = \{(p_i, r_i, w_i),\ i \in N\}$ that is revealed online, and a distribution f.

(1) Construct the preemptive LP-schedule on a single virtual machine m-times faster $(I \mapsto I_m)$.
(2) Draw α randomly from the distribution $f(\alpha)$.
(3) Apply preemptive list-scheduling in order of non-decreasing $t_j(\alpha)$ on the m machines.

Repeating an observation from Schulz and Skutella [22], we see that at any given time, the *order* of the $t_j(\alpha)$ of already released jobs can be found, even if the actual values of $t_j(\alpha)$ are not known. Consider an arbitrary pair of jobs j, k. If $t_j(\alpha_j)$ is known and $t_k(\alpha_k)$ is not known, then clearly $t_j(\alpha_j) \leq t_k(\alpha_k)$. If both α-points are unknown, we can infer that $t_j(\alpha_j) \leq t_k(\alpha_k)$ if and only if $w_j/p_j \geq w_k/p_k$.

It is interesting to note that if step (2) is replaced by "Take $\alpha = 1$" the algorithm becomes a deterministic online algorithm and it coincides with Megow and Schulz's 2-competitive algorithm for $P|r_j, pmtn|\sum_j w_j C_j$ [17]. On the other hand, if f is

taken as the uniform distribution in [0, 1], PASR is also 2-competitive (and this follows as a consequence of the forthcoming analysis).

Let C_j^α denote the completion time of job j in the schedule output by algorithm PASR. Consider job j. Define J as the set of jobs that start before job j in the LP-schedule. For any $k \neq j$, let η_k denote the fraction of job k that is completed in the LP-schedule by time $t_j(0^+)$; note that we are using a slightly different definition for η_k than the definition used in Sects. 3 and 4. Note that $\eta_k = 0, \ \forall k \notin J$. We also have that $t_j(0^+) \geq \sum_{k \in J} \eta_k \frac{p_k}{m}$. Now, define $K_1 = \{k \mid t_k(\alpha) < t_j(0^+)\}$ and $K_2 = \{k \mid t_j(0^+) < t_k(\alpha) < t_j(\alpha)\}$ ($K = K_1 \cup K_2$ is the set of jobs that can preempt job j). Note that jobs $k \in K_2$ preempt job j in the LP-schedule and are all processed in the interval $[t_j(0^+), t_j(\alpha)]$. Consequently, $t_j(\alpha) = t_j(0^+) + \sum_{k \in K_2} \frac{p_k}{m} + \alpha \frac{p_j}{m}$. The following bound is central to our analysis.

Lemma 5.1

$$C_j^\alpha \leq t_j(\alpha) + \left(1 - \frac{\alpha}{m}\right) p_j + \sum_{k \in J, \eta_k \geq \alpha} \left(1 - \frac{\eta_k}{m}\right) \frac{p_k}{m}.$$

Proof We first note that if the LP-schedule is busy, then at least one machine is busy in the schedule defined by PASR. Thus, by time $t_j(0^+)$, the LP-schedule will have processed a total of $\sum_{k \in K_1} \eta_k \frac{p_k}{m}$ and consequently, so will have the schedule defined by algorithm PASR.

We now make some assumptions that can only increase the completion time of job j: (1) Job j has not begun processing in the schedule defined by PASR at time $t_j(0^+)$ and (2) jobs $k \in K_2$ are released at time $t_j(0^+)$ (note that, originally, jobs in K_2 were released sometime in the interval $[t_j(0^+), t_j(\alpha)]$). Under Assumptions (1) and (2), at time $t_j(0^+)$, the amount of *available* processing that remains from $K_1 \cup K_2$ is at most $\sum_{k \in K_2} p_k + \sum_{k \in K_1} (1 - \frac{\eta_k}{m}) p_k$. Since we have m machines, by standard averaging arguments, we have that

$$\begin{aligned}
C_j^\alpha &\leq t_j(0^+) + \sum_{k \in K_2} \frac{p_k}{m} + \sum_{k \in K_1} \left(1 - \frac{\eta_k}{m}\right) \frac{p_k}{m} + p_j \\
&= t_j(0^+) + \sum_{k \in K_2} \frac{p_k}{m} + \sum_{k \in J, \eta_k \geq \alpha} \left(1 - \frac{\eta_k}{m}\right) \frac{p_k}{m} + p_j \\
&= t_j(0^+) + \sum_{k \in K_2} \frac{p_k}{m} + \alpha \frac{p_j}{m} + \left(1 - \frac{\alpha}{m}\right) p_j + \sum_{k \in J, \eta_k \geq \alpha} \left(1 - \frac{\eta_k}{m}\right) \frac{p_k}{m} \\
&= t_j(\alpha) + \left(1 - \frac{\alpha}{m}\right) p_j + \sum_{k \in J, \eta_k \geq \alpha} \left(1 - \frac{\eta_k}{m}\right) \frac{p_k}{m}.
\end{aligned}$$

□

The next lemma generalizes a result by Schulz and Skutella [22].

Lemma 5.2 *Suppose there exists a distribution $f(\alpha)$ and a constant $\gamma_m \in (0, 1)$ such that:*

- $\max_{\alpha\in[0,1]} f(\alpha) \le 1+\gamma_m$.
- $(1-\frac{\eta}{m})\int_0^\eta f(\alpha)d\alpha \le \gamma_m\eta,\ \forall\eta\in[0,1]$.
- $1-\frac{\mathrm{E}\{\alpha\}}{m} \le \frac{1+\gamma_m}{2}$.

Then, Algorithm PASR is $(1+\gamma_m)$*-competitive.*

Proof Using Lemma 5.1, we have that

$$\mathrm{E}\left[C_j^\alpha\right] \le \mathrm{E}\left[t_j(\alpha)\right] + \mathrm{E}\left[\left(1-\frac{\alpha}{m}\right)p_j\right] + \mathrm{E}\left[\sum_{k\in J,\eta_k\ge\alpha}\left(1-\frac{\eta_k}{m}\right)\frac{p_k}{m}\right].$$

We bound each term individually:

$$\begin{aligned}
\mathrm{E}[t_j(\alpha)] &= \mathrm{E}[t_j(0^+) + t_j(\alpha) - t_j(0^+)] \\
&= t_j(0^+) + \int_0^1 f(\alpha)(t_j(\alpha) - t_j(0^+))d\alpha \\
&\le t_j(0^+) + (1+\gamma_m)(M_j - t_j(0^+)) \\
&= (1+\gamma_m)M_j - \gamma_m t_j(0^+); \\
\mathrm{E}\left[\left(1-\frac{\alpha}{m}\right)p_j\right] &= \left(1-\frac{\mathrm{E}[\alpha]}{m}\right)p_j \le (1+\gamma_m)\frac{p_j}{2}; \\
\mathrm{E}\left[\sum_{k\in J,\eta_k\ge\alpha}\left(1-\frac{\eta_k}{m}\right)\frac{p_k}{m}\right] &= \sum_{k\in J}\left(1-\frac{\eta_k}{m}\right)\frac{p_k}{m}\int_0^{\eta_k} f(\alpha)d\alpha \\
&\le \gamma_m\sum_{k\in J}\eta_k\frac{p_k}{m} \le \gamma_m t_j(0^+).
\end{aligned}$$

Summing the terms it follows that

$$\mathrm{E}[C_j^\alpha] \le (1+\gamma_m)M_j - \gamma_m t_j(0^+) + (1+\gamma_m)\frac{p_j}{2} + \gamma_m t_j(0^+) = (1+\gamma_m)\left(M_j + \frac{p_j}{2}\right).$$

Multiplying by w_j and summing over j, we conclude that

$$\begin{aligned}
\mathrm{E}\left[\sum_{j\in N} w_j C_j^\alpha\right] &\le (1+\gamma_m)\sum_{j\in N} w_j(M_j + p_j/2) \\
&= (1+\gamma_m)(Z_R^m(I) + \sum_{j\in N} w_j p_j/2) \le (1+\gamma_m)Z^m(I).
\end{aligned}$$

□

Consider the following distribution for α for the case where we have m machines:

$$f(\alpha) = \begin{cases} \gamma_m \frac{m^2}{(m-\alpha)^2}, & \alpha \in [0, \delta_m] \\ (1+\gamma_m), & \alpha \in (\delta_m, 1], \end{cases}$$

where $\gamma_m = \frac{\delta_m(m-\delta_m)}{m-\delta_m(1-\delta_m)}$ and $\delta_m \in (0, 1]$. Note that for $m = 1$, if we let $\delta_1 = \frac{1}{2}$, then $\gamma_1 = \frac{1}{3}$ and f is exactly the distribution chosen by Schulz and Skutella [22], which gives a $(1+\gamma_1) = \frac{4}{3}$-competitive algorithm. We now show that the best possible choices for δ_m are $\delta_2 = 2(\sqrt{2}-1)$ and $\delta_m = 1$, for all $m \geq 3$. For simplicity we separate the cases $m = 2$ and $m \geq 3$.

In the case $m \geq 3$, we have that $\gamma_m = \frac{m-1}{m}$ and the distribution becomes:

$$f(\alpha) = \gamma_m \frac{m^2}{(m-\alpha)^2}, \quad \alpha \in (0, 1].$$

In the series of lemmas that follow, we show that the above distribution satisfies the conditions of Lemma 5.2, proving Theorem 5.3 (the competitive ratios are plotted in Fig. 1).

Theorem 5.3 *Algorithm PASR is* $(2 - 1/m)$*-competitive for* $m \geq 3$.

Lemma 5.4 *For* $m \geq 3$*, we have that* $\max_{\alpha\in[0,1]} f(\alpha) \leq 1 + \gamma_m$.

Proof Note that $f(\alpha)$ is increasing; consequently,

$$\max_{\alpha\in[0,1]} f(\alpha) = \frac{m}{m-1}.$$

Since $(1+\gamma_m) = 2 - 1/m$, the bound clearly holds for $m \geq 3$. □

We now present a result related to the second requirement of Lemma 5.2.

Lemma 5.5 *For* $m \geq 1$*, we have that*

$$\left(1 - \frac{\eta}{m}\right) \int_0^\eta f(\alpha) d\alpha = \gamma_m \eta, \ \forall \eta \in [0, 1].$$

Proof $\left(1 - \frac{\eta}{m}\right) \int_0^\eta f(\alpha) d\alpha = \left(1 - \frac{\eta}{m}\right) \gamma_m m \frac{\eta}{m-\eta} = \gamma_m \eta$. □

Finally, we address the third requirement of Lemma 5.2.

Lemma 5.6 *For* $m \geq 2$,

$$1 - \frac{\mathrm{E}[\alpha]}{m} \leq \frac{1+\gamma_m}{2}. \tag{9}$$

Proof We first drop the subscript m from γ_m. We may express $\mathrm{E}[\alpha]$ as

$$\mathrm{E}[\alpha] = \gamma m^2 \left(\frac{1}{m-1} + \ln\left(\frac{m-1}{m} \right) \right).$$

Note that

$$1 - \frac{\mathrm{E}[\alpha]}{m} = (m-1)\ln\left(\frac{m}{m-1}\right) \quad \text{and} \quad \frac{1+\gamma}{2} = \frac{2m-1}{2m}.$$

Therefore, it is sufficient to prove that for $m \geq 2$,

$$\ln\left(\frac{m}{m-1}\right) \leq \frac{2m-1}{2m(m-1)}.$$

Letting $x = m/(m-1)$ (note that $m \geq 2 \Leftrightarrow x \in (1, 2]$), it is equivalent to show

$$\ln(x) \leq \frac{x^2-1}{2x}.$$

Enlarging the range of x, we see that the inequality holds with equality at $x = 1$. The inequality would then be true iff the RHS increases faster than the LHS. Taking derivatives, we need to show that $\frac{1}{x} \leq \frac{x^2+1}{2x^2}$, which clearly holds for any real x. □

Consequently, Lemmas 5.2 and 5.4–5.6 give us Theorem 5.3.

Similarly, in the case $m = 2$, we have that $\gamma = \frac{\delta(2-\delta)}{2-\delta(1-\delta)}$ and $\delta = 2(\sqrt{2}-1)$, and the distribution becomes:

$$f(\alpha) = \begin{cases} \gamma \frac{4}{(2-\alpha)^2}, & \alpha \in [0, \delta] \\ (1+\gamma), & \alpha \in (\delta, 1]. \end{cases}$$

In this situation we can again show that the distribution satisfies the conditions of Lemma 5.2, proving the second result of this section.

Theorem 5.7 *Algorithm PASR is* $(1+\gamma)$*-competitive for* $m = 2$*, where* $(1+\gamma) < 1.5225$.

Proof We start by observing that $f(\alpha)$ is increasing in $(0, \delta]$. Then,

$$\max_{\alpha \in [0,\delta]} f(\alpha) = \gamma \frac{4}{(2-\delta)^2} \leq 1 + \gamma,$$

for all $-2(\sqrt{2}+1) \leq \delta \leq 2(\sqrt{2}-1)$, so the first condition in Lemma 5.2 follows.

To see the second requirement of Lemma 5.2, note that for $\eta \leq \delta$, we have that $(1 - \frac{\eta}{2}) \int_0^\eta f(\alpha) d\alpha = (1 - \frac{\eta}{2}) 2\gamma \frac{\eta}{2-\eta} = \gamma\eta$. On the other hand, for $\eta > \delta$,

$$\int_0^\eta f(\alpha) d\alpha = \frac{2\eta - 2\delta + \delta^2 + \delta\eta}{2 - \delta(1-\delta)} \leq \gamma\eta \frac{2}{2-\eta},$$

for the chosen values of γ and δ.

Finally, we address the third requirement of Lemma 5.2.

$$1 - \frac{\mathrm{E}[\alpha]}{2} \leq \frac{1+\gamma}{2}, \tag{10}$$

which follows immediately since $\mathrm{E}[\alpha] = 4\gamma\left(\frac{\delta}{2-\delta} + \ln\left(\frac{2-\delta}{2}\right)\right) + (1+\gamma)\frac{1-\delta^2}{2}$. □

To finish the section we observe that our analysis is tight. Indeed, for $m \geq 3$, setting $\eta = 1$ in the second requirement of Lemma 5.2 implies that $\gamma_m \geq (m-1)/m$. Similarly, for $m = 2$, setting $\eta = \delta_2$ in the second requirement of Lemma 5.2 implies that $\gamma_2 \geq \frac{\delta(2-\delta)}{2-\delta(1-\delta)}$.

6 Computational results

To conclude this paper, we present a computational study of our algorithms when the problem data is randomly generated. To the best of our knowledge, this study is one of the first computational investigations of LP-based scheduling algorithms. Particularly, we believe it is the first that considers the combination of online algorithms, a multiple machine environment and the objective of minimizing the weighted sum of completion times. We are aware of only two other papers that investigate a similar topic. Savelsbergh et al. [19] perform an extensive computational study of a number of heuristics and approximation algorithms for the single machine problem $1|r_j|\sum_j w_j C_j$ (the offline version). As in their work, our results suggest that the practical performance of LP-based scheduling algorithms is much better than what theory predicts. However, it is worth noting that Albers and Schröder [2] perform a computational study of online algorithms for parallel machine scheduling problems with the objective of minimizing the makespan. They experimentally show that online algorithms that perform well on randomly generated data do not necessarily perform well on real-world data.

We let the data for each job be independent realizations of uniformly distributed random variables: $r_i \sim U[0, R]$, $p_i \sim U[0, P]$ and $w_i \sim U[0, W]$, for $i = 1, \ldots, n$.

We now describe our general approach. We first fix the parameters R, P and W and vary m and n. Then, we fix m and n and vary R, P and W. For each set of parameters we run 1000 trials to give the mean, max and standard deviation (presented in this order) of the ratio of the cost of the online algorithm (NAS, NASR or PASR) to the lower bound given in Lemma 2.1.

6.1 NAS

We first fix $R = P = W = 10$ and study the effect of changing m and n.

	$n = 10$	$n = 100$	$n = 500$
$m = 1$	(1.2226, 1.4321, 0.0421)	(1.0283, 1.0433, 0.0019)	(1.0056, 1.0063, 0.0001)
$m = 10$	(1.3275, 1.5293, 0.0559)	(1.1579, 1.1842, 0.0063)	(1.0421, 1.0449, 0.0009)
$m = 25$	(1.3308, 1.5548, 0.0614)	(1.2613, 1.2866, 0.0076)	(1.0871, 1.0943, 0.0017)

Next, we fix $P = W = 10$ and let R depend on n; specifically, we let $R = n$ for $n \in \{10, 100, 500\}$. We also vary m as before: $m \in \{1, 10, 25\}$.

	$R = n = 10$	$R = n = 100$	$R = n = 500$
$m = 1$	(1.2226, 1.4321, 0.0421)	(1.0463, 1.0958, 0.0111)	(1.0131, 1.0252, 0.0031)
$m = 10$	(1.3275, 1.5293, 0.0559)	(1.0526, 1.0677, 0.0045)	(1.0112, 1.0128, 0.0005)
$m = 25$	(1.3308, 1.5548, 0.0614)	(1.0521, 1.0674, 0.0045)	(1.0110, 1.0123, 0.0004)

Finally, we fix $m = 10$ and $n = 100$ (the middle case) and vary R, P and W in the following set $\{1, 10\}$; we present the eight results below.

	$(R, P) = (1, 1)$	$(R, P) = (1, 10)$
$W = 1$	(1.1057, 1.1139, 0.0026)	(1.1579, 1.1776, 0.0053)
$W = 10$	(1.1057, 1.1157, 0.0025)	(1.1583, 1.1784, 0.0054)

	$(R, P) = (10, 1)$	$(R, P) = (10, 10)$
$W = 1$	(1.0799, 1.0915, 0.0040)	(1.1578, 1.1876, 0.0061)
$W = 10$	(1.0799, 1.0928, 0.0041)	(1.1581, 1.1816, 0.0061)

6.2 NASR

We first fix $R = P = W = 10$ and study the effect of changing m and n. The expected competitive ratio for $m = 1$, $m = 10$ and $m = 25$ are 1.6853, 1.9673 and 1.9869, respectively.

	$n = 10$	$n = 100$	$n = 500$
$m = 1$	(1.2140, 1.5575, 0.0569)	(1.0336, 1.0429, 0.0026)	(1.0075, 1.0083, 0.0002)
$m = 10$	(1.3186, 1.5094, 0.0574)	(1.1536, 1.1849, 0.0065)	(1.0415, 1.0449, 0.0009)
$m = 25$	(1.3331, 1.6013, 0.0607)	(1.2574, 1.2835, 0.0081)	(1.0863, 1.0918, 0.0016)

Next, we fix $P = W = 10$ and let R depend on n; specifically, we let $R = n$ for $n \in \{10, 100, 500\}$. We also vary m as before: $m \in \{1, 10, 25\}$.

	$R = n = 10$	$R = n = 100$	$R = n = 500$
$m = 1$	(1.2140, 1.5575, 0.0569)	(1.0449, 1.0903, 0.0098)	(1.0121, 1.0238, 0.0030)
$m = 10$	(1.3186, 1.5094, 0.0574)	(1.0517, 1.0717, 0.0044)	(1.0109, 1.0124, 0.0004)
$m = 25$	(1.3331, 1.6013, 0.0607)	(1.0516, 1.0685, 0.0045)	(1.0109, 1.0123, 0.0005)

Finally, we fix $m = 10$ and $n = 100$ and vary R, P and W in the following set $\{1, 10\}$; we present the eight results below.

	$(R, P) = (1, 1)$	$(R, P) = (1, 10)$
$W = 1$	(1.1080, 1.1190, 0.0028)	(1.1553, 1.1711, 0.0053)
$W = 10$	(1.1079, 1.1207, 0.0027)	(1.1551, 1.1739, 0.0052)

	$(R, P) = (10, 1)$	$(R, P) = (10, 10)$
$W = 1$	(1.0808, 1.0945, 0.0046)	(1.1535, 1.1766, 0.0065)
$W = 10$	(1.0812, 1.0951, 0.0047)	(1.1533, 1.1772, 0.0066)

6.3 PASR

We first fix $R = P = W = 10$ and study the effect of changing m and n. The expected competitive ratio for $m = 1$, $m = 10$ and $m = 25$ are 1.3333, 1.8961 and 1.9595, respectively.

	$n = 10$	$n = 100$	$n = 500$
$m = 1$	(1.0887, 1.5016, 0.0650)	(1.0015, 1.0091, 0.0013)	(1.0000, 1.0004, 0.0000)
$m = 10$	(1.2678, 1.3957, 0.0461)	(1.0430, 1.0756, 0.0110)	(1.0015, 1.0038, 0.0006)
$m = 25$	(1.3081, 1.4914, 0.0559)	(1.1515, 1.2112, 0.0160)	(1.0117, 1.7732, 0.0242)

Next, we fix $P = W = 10$ and let R depend on n; specifically, we let $R = n$ for $n \in \{10, 100, 500\}$. We also vary m as before: $m \in \{1, 10, 25\}$.

	$R = n = 10$	$R = n = 100$	$R = n = 500$
$m = 1$	(1.0887, 1.5016, 0.0650)	(1.0107, 1.0595, 0.0071)	(1.0028, 1.0189, 0.0019)
$m = 10$	(1.2678, 1.3957, 0.0461)	(1.0426, 1.0540, 0.0037)	(1.0090, 1.0105, 0.0004)
$m = 25$	(1.3081, 1.4914, 0.0559)	(1.0486, 1.0634, 0.0040)	(1.0102, 1.0114, 0.0004)

Finally, we fix $m = 10$ and $n = 100$ and vary R, P and W in the following set $\{1, 10\}$; we present the eight results below.

	$(R, P) = (1, 1)$	$(R, P) = (1, 10)$
$W = 1$	(1.0037, 1.3452, 0.0108)	(1.0065, 1.0095, 0.0007)
$W = 10$	(1.0034, 1.0048, 0.0003)	(1.0065, 1.0094, 0.0007)

	$(R, P) = (10, 1)$	$(R, P) = (10, 10)$
$W = 1$	(1.0220, 1.0662, 0.0029)	(1.0433, 1.5628, 0.0199)
$W = 10$	(1.0219, 1.0316, 0.0023)	(1.0436, 1.0815, 0.0117)

6.4 Observations

- For all three algorithms, as m increases, the mean and max increase.
- For all three algorithms, as n increases, the mean and max decrease.
- Statistically, it seems that NAS and NASR are comparable.
- Over 1000 trials, the max ratio of NASR never exceeded the expected competitive ratio.
- Over 1000 trials, the max ratio of PASR did exceed the expected competitive ratio (e.g., $m = 1, n = 10$).
- Fixing P, W and letting $R = n$ vary shows that as R increases, the mean competitive ratio *decreases* for all values of m tested.
- Varying $W \in (0, m]$ (and keeping other parameters constant) does not significantly affect the performance of the algorithms.

Acknowledgements We thank Andreas Schulz and an anonymous referee for carefully reading this manuscript and for several pointers to relevant literature. We specially thank another anonymous referee for pointing out an error in a preliminary version of Theorem 5.3. Finally, we thank Vladimir Barzov for implementing the algorithms designed in this paper. The research of the first author was partially supported by CONICYT through grants FONDECYT 1060035 and ACT08.

Appendix A: Technical details concerning Eq. (5)

In this appendix, we first discuss Eq. (5):

$$\ln\left(1+\frac{1}{m}-\frac{\gamma}{m}\right)+\frac{\gamma}{m} = \frac{e^{(-\gamma/m)}\left(1+\frac{1}{m}-\frac{\gamma}{m}-e^{(-\gamma/m)}\right)\left(me^{(\gamma/m)}-\gamma e^{(\gamma/m)}+\frac{1}{m}e^{(\gamma/m)}+1-m\right)}{1+\frac{1}{m}-\frac{\gamma}{m}}$$

and we then discuss some details concerning δ_m and c_m.

Existence of $\gamma \in (0, 1)$

We show that, for any finite m, there exists $\gamma \in (0, 1)$ that satisfies Eq. (5). Let

$$l(\gamma) = \ln\left(1+\frac{1}{m}-\frac{\gamma}{m}\right)+\frac{\gamma}{m}$$

and

$$r(\gamma) = \frac{e^{(-\gamma/m)}\left(1+\frac{1}{m}-\frac{\gamma}{m}-e^{(-\gamma/m)}\right)\left(me^{(\gamma/m)}-\gamma e^{(\gamma/m)}+\frac{1}{m}e^{(\gamma/m)}+1-m\right)}{1+\frac{1}{m}-\frac{\gamma}{m}}.$$

We have that $l(0) = \ln(1+\frac{1}{m})$ and $r(0) = \frac{1}{m}$; note that for finite m, $l(0) < r(0)$. For $\gamma = 1$, we have that $l(1) = \frac{1}{m}$ and $r(1) = e^{(-1/m)}\left(1-e^{(-1/m)}\right)\left((e^{(1/m)}-1)(m-1)+\frac{1}{m}e^{(1/m)}\right)$. We next show that $l(1) > r(1)$, which is equivalent to showing $(e^{(1/m)}+e^{(-1/m)}-2)(m-1) < \frac{1}{m}$. To see the latter, we bound $e^{(-1/m)}$ using the standard Taylor series bound $e^{-x} \le 1-x+\frac{x^2}{2}$ and bound $e^{(1/m)}$ by using

$$\begin{aligned} e^x &= 1+x+\frac{x^2}{2!}+\frac{x^3}{3!}+\cdots < 1+x+\frac{x^2}{2}\left(1+x+x^2+\cdots\right) \\ &= 1+x+\frac{x^2}{2}\left(\frac{1}{1-x}\right). \end{aligned}$$

Substituting the values we obtain $l(1) > r(1)$, as desired. Since both $l(\gamma)$ and $r(\gamma)$ are continuous functions of γ, it is clear that there exists a value of $\gamma \in (0, 1)$ such that $l(\gamma) = r(\gamma)$.

6.5 Uniqueness of $\gamma \in (0, 1)$

Define $x = (1 + \frac{1}{m} - \frac{\gamma}{m})e^{(\gamma/m)} \geq 0$; Eq. (5) can then be written as

$$\frac{x \ln x}{x - 1} = m - \gamma + \frac{1}{m} - (m - 1)e^{-(\gamma/m)}.$$

Let $l(\gamma) = \frac{x \ln x}{x-1}$ and $r(\gamma) = m - \gamma + \frac{1}{m} - (m - 1)e^{-(\gamma/m)}$. Note that

$$\frac{dl(\gamma)}{d\gamma} = \frac{dl(\gamma)}{dx}\frac{dx}{d\gamma} = \frac{(x - 1 - \ln x)}{(x - 1)^2}\frac{(1 - \gamma)e^{(\gamma/m)}}{m^2} \geq 0$$
$$\frac{dr(\gamma)}{d\gamma} = -1 + \frac{m - 1}{m}e^{-(\gamma/m)} < 0.$$

Consequently, there is an unique solution to Eq. (5).

6.6 Feasibility of $f(\alpha)$: $\delta_m \in (0, 1),\ \forall m \geq 1$

Using the notation and analysis from above, we have that δ_m satisfies: $0 < ml(0) < \delta_m < ml(1) = 1$.

6.7 Calculation showing $\lim_{m\to\infty} \delta_m = 1$

Observing that $ml(0) = m \ln (1 + \frac{1}{m})$ and applying l'Hôpital's Rule,

$$\lim_{m\to\infty} m \ln (1 + \frac{1}{m}) = \lim_{m\to\infty} \frac{\left(\frac{1}{1+\frac{1}{m}}\right)\left(-\frac{1}{m^2}\right)}{-\frac{1}{m^2}} = 1.$$

6.8 Calculation showing $\lim_{m\to\infty} c_m = 1$

Note that $\gamma \in (0, 1)$; we have that

$$\begin{aligned}\lim_{m\to\infty} c_m &= \lim_{m\to\infty} \frac{1/m}{e^{(\gamma/m)}\left(1 + \frac{1}{m} - \frac{\gamma}{m}\right) - 1} \\ &= \lim_{m\to\infty} \frac{-1/m^2}{e^{(\gamma/m)}\left(\frac{\gamma}{m^2} - \frac{1}{m^2}\right) + \left(1 + \frac{1}{m} - \frac{\gamma}{m}\right)e^{(\gamma/m)}\left(-\frac{\gamma}{m^2}\right)} \text{ (by l'Hôpital's Rule)} \\ &= 1.\end{aligned}$$

References

1. Afrati, F., Bampis, E., Chekuri, C., Karger, D., Kenyon, C., Khanna, S., Milis, I., Queyranne, M., Skutella, M., Stein C., Sviridenko, M.: Approximation schemes for minimizing average weighted completion time with release dates. In: Proceedings of the 40th IEEE Symposium on Foundations of Computer Science (FOCS), pp. 32–43 (1999)
2. Albers, S., Schröder, B.: An experimental study of online scheduling algorithms. J. Exp. Algorithmics **7**, 3 (2002)
3. Anderson, E.J., Potts, C.N.: On-line scheduling of a single machine to minimize total weighted completion time. Math. Oper. Res. **29**, 686–697 (2004)
4. Chakrabarti, S., Phillips, C., Schulz, A.S., Shmoys, D.B., Stein, C., Wein, J.: Improved scheduling algorithms for minsum criteria," in Automata, Languages and Programming (ICALP), LNCS, vol. 1099, pp. 646–657. Springer, Heidelberg (1996)
5. Chekuri, C., Motwani, R., Natarajan, B., Stein, C.: Approximation techniques for average completion time scheduling. SIAM J. Comput. **31**, 146–166 (2001)
6. Chou, M.C., Queyranne, M., Simchi-Levi, D.: The asymptotic performance ratio of an on-line algorithm for uniform parallel machine scheduling with release dates. Math. Program. **106**, 137–157 (2006)
7. Sousa, J.P.: Time Indexed Formulations of Non-Preemptive Single-Machine Scheduling Problems. PhD thesis, Université Catholique de Louvain (1989)
8. Eastman, W.L., Even, S., Isaacs, I.M.: Bounds for the optimal scheduling of n jobs on m processors. Manage. Sci. **11**, 268–279 (1964)
9. Goemans, M.X.: A supermodular relaxation for scheduling with release dates. In: Proceedings of the 5th Integer Programming and Combinatorial Optimization Conference (IPCO), LNCS, vol. 1084, pp. 288–300. Springer, Heidelberg (1996)
10. Goemans, M.X.: Improved approximation algorithms for scheduling with release dates. In: Proceedings of the 8th Annual ACM–SIAM Symposium on Discrete Algorithms (SODA), New Orleans, LA, 1997. ACM, New York, 1997, pp. 591–598 (1997)
11. Goemans, M., Queyranne, M., Schulz, A.S., Skutella, M., Wang, Y.: Single machine scheduling with release dates. SIAM J. Discrete Math. **15**, 165–192 (2002)
12. Graham, R.L., Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G.: Optimization and approximation in deterministic sequencing and scheduling: a survey. Ann. Discrete Math. **5**, 287–326 (1979)
13. Hall, L.A., Schulz, A.S., Shmoys, D.B., Wein, J.: Scheduling to minimize average completion time: off-line and on-line approximation algorithms. Math. Oper. Res. **22**, 513–544 (1997)
14. Hall, L.A., Shmoys, D.B., Wein, J.: Scheduling to minimize average completion time: off-line and on-line approximation algorithms. In: Proceedings of the 7th Annual ACM–SIAM Symposium on Discrete Algorithms, pp. 142–151 (1997)
15. Hoogeveen, J.A., Vestjens, A.P.A.: Optimal on-line algorithms for single-machine scheduling. In: Proceedings of the 5th Integer Programming and Combinatorial Optimization Conference (IPCO), LNCS, vol. 1084, pp. 404–414. Springer, Heidelberg (1996)
16. Kovács, A., Beck, J.C.: A global constraint for total weighted completion time. In: Proceedings of the 4th International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (LNCS 4510), Brussels, pp. 112–126. Springer, Heidelberg (2007)
17. Megow, N., Schulz, A.S.: On-line scheduling to minimize average completion time revisited. Oper. Res. Lett **32**, 485–490 (2004)
18. Phillips, C., Stein, C., Wein, J.: Minimizing average completion time in the presence of release dates. Math. Program. **82**, 199–223 (1998)
19. Savelsbergh, M.W.P., Uma, R.N., Wein, J.: An experimental study of LP-based approximation algorithms for scheduling problems. INFORMS J. Comput. **17**, 123–136 (2005)
20. Schulz, A.S.: New old algorithms for stochastic scheduling. In: Albers, S., Möhring, R.H., Pflug, G.C., Schultz, R. (eds.) Algorithms for Optimization with Incomplete Information, Dagstuhl Seminar Proceedings 05031, Schloss Dagstuhl, Germany (2005)
21. Schulz, A.S., Skutella, M.: Scheduling unrelated machines by randomized rounding. SIAM J. Discrete Math. **15**, 450–469 (2002)
22. Schulz, A.S., Skutella, M.: The power of α-points in preemptive single machine scheduling. J. Sched. **5**, 121–133 (2002)
23. Seiden, S.: A guessing game and randomized online algorithms. In: Proceedings of the 32nd ACM Symposium on Theory of Computing, pp. 592–601 (2000)

24. Sitters, R.: Complexity and approximation in routing and scheduling. PhD Thesis, Eindhoven University of Technology, The Netherlands (2004)
25. Skutella, M.: List scheduling in order of α-points on a single machine. Lecture Notes in Computer Science, vol. 3484, pp. 250–291 (2006)
26. Smith, W.E.: Various optimizers for single-stage production. Nav. Res. Logist. Q. **3**, 59–66 (1956)
27. Stougie, L., Vestjens, A.P.A.: Randomized algorithms for on-line scheduling problems: how low can't you go?. Oper. Res. Lett. **30**, 89–96 (2002)
28. Vestjens, A.P.A.: Online machine scheduling. PhD Thesis, Eindhoven University of Technology, The Netherlands (1997)