Scheduling in the Random-Order Model

Susanne Albers
Department of Computer Science, Technical University of Munich, Germany
albers@in.tum.de

Maximilian Janke
Department of Computer Science, Technical University of Munich, Germany
maximilian.janke@in.tum.de

Abstract

Makespan minimization on identical machines is a fundamental problem in online scheduling. The goal is to assign a sequence of jobs to m identical parallel machines so as to minimize the maximum completion time of any job. Already in the 1960s, Graham showed that *Greedy* is $(2 - 1/m)$-competitive [18]. The best deterministic online algorithm currently known achieves a competitive ratio of 1.9201 [14]. No deterministic online strategy can obtain a competitiveness smaller than 1.88 [34].

In this paper, we study online makespan minimization in the popular random-order model, where the jobs of a given input arrive as a random permutation. It is known that *Greedy* does not attain a competitive factor asymptotically smaller than 2 in this setting [32]. We present the first improved performance guarantees. Specifically, we develop a deterministic online algorithm that achieves a competitive ratio of 1.8478. The result relies on a new analysis approach. We identify a set of properties that a random permutation of the input jobs satisfies with high probability. Then we conduct a worst-case analysis of our algorithm, for the respective class of permutations. The analysis implies that the stated competitiveness holds not only in expectation but with high probability. Moreover, it provides mathematical evidence that job sequences leading to higher performance ratios are extremely rare, pathological inputs. We complement the results by lower bounds for the random-order model. We show that no deterministic online algorithm can achieve a competitive ratio smaller than 4/3. Moreover, no deterministic online algorithm can attain a competitiveness smaller than 3/2 with high probability.

2012 ACM Subject Classification Theory of computation → Online algorithms

Keywords and phrases Scheduling, makespan minimization, online algorithm, competitive analysis, lower bound, random-order

Digital Object Identifier 10.4230/LIPIcs.ICALP.2020.68

Category Track A: Algorithms, Complexity and Games

Related Version https://arxiv.org/abs/2006.00386

Funding Work supported by the European Research Council, Grant Agreement No. 691672, project APEG.

1 Introduction

We study one of the most basic scheduling problems. Consider a sequence of jobs $\mathcal{J} = J_1, \ldots, J_n$ that has to be assigned to m identical parallel machines. Each job J_t has an individual processing time p_t, $1 \leq t \leq n$. Preemption of jobs is not allowed. The goal is to minimize the *makespan*, i.e. the maximum completion time of any job in the constructed schedule. Both the offline and online variants of this problem have been studied extensively, see e.g. [4, 11, 14, 18, 20, 33] and references therein.

We focus on the online setting, where jobs arrive one by one. Whenever a job J_t is presented, its processing time p_t is revealed. The job has to be scheduled immediately on one of the machines without knowledge of any future jobs J_s, with $s > t$. Given a job sequence $\mathcal{J}$,

let $A(\mathcal{J})$ denote the makespan of an online algorithm A on $\mathcal{J}$. Let $OPT(\mathcal{J})$ be the optimum makespan. A deterministic online algorithm A is *c-competitive* if $A(\mathcal{J}) \leq c \cdot OPT(\mathcal{J})$ holds for all $\mathcal{J}$ [38]. The best competitive ratio that can be achieved by deterministic online algorithms is in the range $[1.88, 1.9201]$. No randomized online algorithm is known that beats deterministic ones, for general m.

In this paper we investigate online makespan minimization in the random-order model. Here an input instance / job sequence is chosen by an adversary. Then a random permutation of the input elements / jobs arrives. The random-order model was considered by Dynkin [10] and Lindley [28] for the secretary problem. Over the last years the framework has received quite some research interest and many further problems have been studied. These include generalized secretary problems [2, 3, 13, 27, 28], the knapsack problem [2, 27], bin packing [25], facility location [30], matching problems [16, 21, 29], packing LPs [26] and convex optimization [19].

We present an in-depth study of online makespan minimization in the random-order model. As a main contribution we devise a new deterministic online algorithm that achieves a competitive ratio of 1.8478. After almost 20 years this is the first progress for the pure online setting, where an algorithm does not resort to extra resources in handling a job sequence.

Previous work. We review the most important results relevant to our work and first address the standard setting where an online algorithm must schedule an arbitrary, worst-case job sequence. Graham in 1966 showed that the famous *Greedy* algorithm, which assigns each job to a least loaded machine, is $(2 - \frac{1}{m})$-competitive. Using new deterministic strategies the competitiveness was improved in a series of papers. Galambos and Woeginger [15] gave an algorithm with a competitive ratio of $(2 - \frac{1}{m} - \epsilon_m)$, where ϵ_m tends to 0 as $m \to \infty$. Bartal et al. [4] devised a 1.986-competitive algorithm. The bound was improved to 1.945 [22] and 1.923 [1]. Fleischer and Wahl [14] presented an algorithm that attains a competitive ratio of 1.9201 as $m \to \infty$. Chen et al. [7] gave an algorithm whose competitiveness is at most $1 + \varepsilon$ times the best possible factor, but no explicit bound was provided. Lower bounds on the competitive ratio of deterministic online algorithms were shown in [1, 5, 12, 17, 34, 35]. For general m, the bound was raised from 1.707 [12] to 1.837 [5] and 1.854 [17]. Rudin [34] showed that no deterministic strategy has a competitiveness smaller than 1.88.

For randomized online algorithms, there is a significant gap between the best known upper and lower bounds. For $m = 2$ machines, Bartal et al. [4] presented an algorithm that achieves an optimal competitive ratio of 4/3. To date, there exists no randomized algorithm whose competitiveness is smaller than the deterministic lower bound, for general m. The best known lower bound on the performance of randomized online algorithms tends to $e/(e-1) \approx 1.581$ as $m \to \infty$ [6, 37].

Recent research on makespan minimization has examined settings where an online algorithm is given extra resources when processing a job sequence. Specifically, an algorithm might have a buffer to reorder the incoming job sequence [11, 24] or is allowed to migrate jobs [36]. Alternatively, an algorithm has information on the job sequence [8, 9, 23, 24], e.g. it might know the total processing time of the jobs or even the optimum makespan.

In the random-order model only one result is known for makespan minimization on identical machines. Osborn und Torng [32] showed that *Greedy* does not achieve a competitive ratio smaller than 2 as $m \to \infty$. Recently Molinaro [31] studied online load balancing with the objective to minimize the l_p-norm of the machine loads. He considers a general scenario with machine-dependent job processing times. For makespan minimization he presents an algorithm that, in the worst case, is $O(\log m/\varepsilon)$-competitive and, in the random-order model, has an expected makespan of $(1 + \varepsilon)OPT(\mathcal{J}) + O(\log m/\varepsilon)$, for any $\varepsilon \in (0, 1]$.

Our contribution. We investigate online makespan minimization in the random-order model, a sensible and widely adopted input model to study algorithms beyond the worst case. Specifically, we develop a new deterministic algorithm that achieves a competitive ratio of 1.8478 as $m \to \infty$. This is the first improved performance guarantee in the random-order model. The competitiveness is substantially below the best known ratio of 1.9201 in the worst-case setting and also below the corresponding lower bound of 1.88 in that framework.

A new feature of our algorithm is that it schedules an incoming job on one of three candidate machines in order to maintain a certain load profile. The best strategies in the worst-case setting use two possible machines, and it is not clear how to take advantage of additional machines in that framework. The choice of our third, extra machine is quite flexible: An incoming job is placed either on a least loaded, a heavily loaded or – as a new option – on an intermediate machine. The latter one is the $(h+1)$-st least loaded machine, where h may be any integer with $h \in \omega(1)$ and $h \in o(\sqrt{m})$.

When assigning a job to a machine different from the least loaded one, an algorithm has to ensure that the resulting makespan does not exceed c times the optimum makespan, for the targeted competitive ratio c. All previous strategies in the literature lower bound the optimum makespan by the current average load on the machines. Our new algorithm works with a refined lower bound that incorporates the processing times of the largest jobs seen so far. The lower bound is obvious but has not been employed by previous algorithms.

The analysis of our algorithm proceeds in two steps. First we define a class of *stable job sequences*. These are sequences that reveal information on the largest jobs as processing volume is scheduled. More precisely, once a certain fraction of the total processing volume $\sum_{t=1}^{n} p_t$ has arrived, one has a good estimate on the h-th largest job and has encountered a certain number of the $m+1$ largest jobs in the input. The exact parameters have to be chosen carefully.

We prove that with high probability, a random permutation of a given input of jobs is stable. We then conduct a worst-case analysis of our algorithm on stable sequences. Using their properties, we show that if the algorithm generates a flat schedule, like *Greedy*, and can be hurt by a huge job, then the input must contain many large jobs so that the optimum makespan is also high. A new ingredient in the worst-case analysis is the processing time of the h-th largest job in the input. We will relate it to machine load in the schedule and to the processing time of the $(m+1)$-st largest job; twice the latter value is a lower bound on the optimum makespan.

The analysis implies that the competitive ratio of 1.8478 holds with high probability. Input sequences leading to higher performance ratios are extremely rare. We believe that our analysis approach might be fruitful in the study of other problems in the random-order model: Identify properties that a random permutation of the input elements satisfies with high probability. Then perform a worst-case analysis.

Finally in this paper we devise lower bounds for the random-order model. We prove that no deterministic online algorithm achieves a competitive ratio smaller than 4/3. Moreover, if a deterministic online algorithm is c-competitive with high probability, then $c \geq 3/2$.

2 Strong competitiveness in the random-order model

We define competitiveness in the random-order model and introduce a stronger measure of competitiveness that implies high-probability bounds. Recall that traditionally a deterministic online algorithm A is c-competitive if $A(\mathcal{J}) \leq c \cdot OPT(\mathcal{J})$ holds for all job sequences $\mathcal{J} = J_1, \ldots, J_n$. We will refer to this worst-case model also as the *adversarial model*.

In the *random-order model* a job sequence $\mathcal{J} = J_1, \ldots, J_n$ is given, which may be specified by an adversary. (Alternatively, a set of jobs could be specified.) Then a random permutation of the jobs arrives. We define the expected cost / makespan of a deterministic online algorithm. Let S_n be the permutation group of the integers from 1 to n, which we consider a probability space under the uniform distribution, i.e. each permutation in S_n is chosen with probability $1/n!$. Given $\sigma \in S_n$, let $\mathcal{J}^\sigma = J_{\sigma(1)}, \ldots, J_{\sigma(n)}$ be the *job sequence permuted by* σ. The expected makespan of A on $\mathcal{J}$ in the random-order model is $A^{\mathrm{rom}}(\mathcal{J}) = \mathbf{E}_{\sigma \sim S_n}[A(\mathcal{J}^\sigma)] = \frac{1}{n!} \sum_{\sigma \in S_n} A(\mathcal{J}^\sigma)$. The algorithm A is *c-competitive in the random-order model* if $A^{\mathrm{rom}}(\mathcal{J}) \leq c \cdot OPT(\mathcal{J})$ holds for all job sequences $\mathcal{J}$.

We next define the notion of a deterministic online algorithm A being *nearly c-competitive*. The second condition in the following definition requires that the probability of A not meeting the desired performance ratio must be arbitrarily small as m grows and a random permutation of a given job sequence arrives. The subsequent Lemma 2 states that a nearly c-competitive algorithm is c-competitive in the random-order model.

▶ **Definition 1.** *A deterministic online algorithm A is called* nearly c-competitive *if the following two conditions hold.*

- *The algorithm A achieves a constant competitive ratio in the adversarial model.*
- *For every $\varepsilon > 0$, there exists an $m(\varepsilon)$ such that for all machine numbers $m \geq m(\varepsilon)$ and all job sequences $\mathcal{J}$ there holds $\mathbf{P}_{\sigma \sim S_n}[A(\mathcal{J}^\sigma) \geq (c+\varepsilon)OPT(\mathcal{J})] \leq \varepsilon$.*

▶ **Lemma 2.** *If a deterministic online algorithm is nearly c-competitive, then it is c-competitive in the random-order model as $m \to \infty$.*

Proof. Let C be the constant such that A is C-competitive in the adversarial model. We may assume that $C > c$. Given $0 < \delta \leq C - c$, we show that there exists an $m(\delta)$ such that, for all $m \geq m(\delta)$, we have $A^{\mathrm{rom}}(\mathcal{J}) \leq (c+\delta)OPT(\mathcal{J})$ for every job sequences $\mathcal{J}$. Let $\varepsilon = \delta/(C-c+1)$. Since A is nearly c-competitive, there exists an $m(\varepsilon)$ such that, for all $m \geq m(\varepsilon)$ and all inputs $\mathcal{J}$, there holds $P_\varepsilon(\mathcal{J}) = \mathbf{P}_{\sigma \sim S_n}[A(\mathcal{J}^\sigma) \geq (c+\varepsilon)OPT(\mathcal{J})] \leq \varepsilon$. Set $m(\delta) = m(\varepsilon)$. We obtain

$$\begin{aligned} A^{\mathrm{rom}}(\mathcal{J}) &\leq (1 - P_\varepsilon(\mathcal{J}))(c+\varepsilon)OPT(\mathcal{J}) + P_\varepsilon(\mathcal{J}) \cdot C \cdot OPT(\mathcal{J}) \\ &\leq ((1-\varepsilon)(c+\varepsilon) + \varepsilon C)OPT(\mathcal{J}) \\ &\leq (c + \varepsilon(C-c+1))OPT(\mathcal{J}) \\ &= (c+\delta)OPT(\mathcal{J}). \end{aligned}$$

◀

3 Description of the new algorithm

The deficiency of *Greedy* is that it tends to generate a flat, balanced schedule in which all the machines have approximately the same load. An incoming large job can then enforce a high makespan relative to the optimum one. It is thus crucial to try to avoid flat schedules and maintain steep schedules that exhibit a certain load imbalance among the machines.

However, in general, this is futile. Consider a sequence of m identical jobs with a processing time of, say, P_{m+1} (refering to the size of the $(m+1)$-st largest job in an input). Any online algorithm that is better than 2-competitive must schedule these m jobs on separate machines, obtaining the flattest schedule possible. An incoming even larger job of processing time $p_{\max}$ will now enforce a makespan of $P_{m+1} + p_{\max}$. Observe that $\mathrm{OPT} \geq \max\{2P_{m+1}, p_{\max}\}$ since there must be one machine containing two jobs. In particular $P_{m+1} + p_{\max} \leq 1.5\mathrm{OPT}$. Hence sensible online algorithms do not perform badly on this sequence.

This example summarizes the quintessential strategy of online algorithms that are good on all sequences: Ensure that in order to create a schedule that is very flat, i.e. such that all machines have high load λ, the adversary must present m jobs that all are large relative to λ. In order to exploit this very flat schedule and cause a high makespan the adversary needs to follow up with yet another large job. But with these $m+1$ jobs, the optimum scheduler runs into the same problem as in the example: Of the $m+1$ large jobs, two have to be scheduled on the same machine. Thus the optimum makespan is high, compensating to the high makespan of the algorithm.

Effectively realizing the aforementioned strategy is highly non-trivial. In fact it is the central challenge in previous works on adversarial makespan minimization that improve upon *Greedy* [1, 4, 14, 15, 22]. These works gave us clear notions of how to avoid flat schedules, which form the basis for our approaches. Instead of simply rehashing these ideas, we want to outline next how we profit from random-order arrival in particular.

3.1 How random-order arrival helps

The first idea to profit from random-order arrival addresses the lower bound on OPT sophisticated online algorithms need. In the literature only the current average load has been considered, but under random-order arrival another bound comes to mind: The largest job seen so far. In order for an algorithm to perform badly, a large job needs to come close to the end of the sequence. Under random-order arrival, it is equally likely for such a job to arrive similarly close to the beginning of the sequence. In this case, the algorithm knows a better lower bound for OPT. The main technical tool will be our *Load Lemma*, which allows us to relate what a job sequence should reveal early from an analysis perspective to the actual fraction of jobs scheduled. This idea does not work for worst-case orders since they tend to order jobs by increasing processing times.

Recall that the general challenge of our later analysis will be to establish that there had to be m large jobs once the schedule gets very flat. In classical analyses, which consider worst-case orders, these jobs appear with increasing density towards the end of the sequence. In random orders this is unlikely, which can be exploited by the algorithm.

The third idea improves upon the first idea. Suppose, that we were to modify our algorithm such that it could handle one very large job arriving close to the end of the sequence. In fact, assume that it could only perform badly when confronted with h very large jobs. We can then disregard any sequence which contains fewer such jobs. Recall that the first idea requires one very large job to arrive sufficiently close to the beginning. Now, as h grows, the probability of the latter event grows as well and approaches 1. This will not only improve our competitive ratio tremendously, it also allows us to adhere to the stronger notion of nearly competitiveness introduced in Section 2. Let us discuss how such a modification is possible: The first step is to design our algorithm in a way that it is reluctant to use the h least loaded machines. Intuitively, if the algorithm tries to retain machines of small load it will require very large jobs to fill them. In order to force these filling jobs to actually be large enough, our algorithm needs to use a very high lower bound for OPT. In fact, here it uses another lower bound for the optimum makespan, $2P^t_{m+1}$, twice the $(m+1)$-st largest job seen so far at time t. Common analysis techniques can only make predictions about P^t_{m+1} at the very end of the sequence. It requires very subtle use of the random-order model to work around this.

3.2 Formal definition

Formally our algorithm ALG is nearly c-competitive, where c is the unique real root of the polynomial $Q[x] = 4x^3 - 14x^2 + 16x - 7$, i.e.

$$c = \frac{7 + \sqrt[3]{28 - 3\sqrt{87}} + \sqrt[3]{28 + 3\sqrt{87}}}{6} < 1.8478.$$

Given $\mathcal{J}$, ALG is presented with a job sequence/permutation $\mathcal{J}^\sigma = J_{\sigma(1)}, \ldots, J_{\sigma(n)}$ that must be scheduled in this order. Throughout the scheduling process ALG always maintains a list of the machines sorted in non-increasing order of current *load*. At any time the load of a machine is the sum of the processing times of the jobs already assigned to it. After ALG has processed the first t jobs $J_{\sigma(1)}, \ldots, J_{\sigma(t)}$, let $M_1^t, \ldots, M_m^t$ be any ordering of the m machines according to non-increasing load. More specifically, let l_j^t denote the load of machine M_j^t. Then $l_1^t \geq \ldots \geq l_m^t$ and l_1^t is the makespan of the current schedule.

ALG places each incoming job $J_{\sigma(t)}$, $1 \leq t \leq n$, on one of three candidate machines. The choice of one machine, having an intermediate load, is flexible. Let $h = h(m)$ be an integer with $h(m) \in \omega(1)$ and $h(m) \in o(\sqrt{m})$. We could use e.g. $h(m) = \lfloor \sqrt[3]{m} \rfloor$ or $h(m) = \lfloor \log m \rfloor$. Let

$$i = \lceil (2c - 3)m \rceil + h \approx 0.6956m.$$

ALG will assign the incoming job to the machine with the smallest load, the $(h+1)$-st smallest load or the i-th largest load.

When scheduling a job on a machine that is different from the least loaded one, an algorithm has to ensure that the resulting makespan does not exceed c^* times the optimum makespan, where c^* is the desired competitiveness. All previous algorithms lower bound the optimum makespan by the current average machine load. Algorithm ALG works with a refined lower bound that incorporates the processing time of the largest job and twice the processing time of the $(m+1)$-st largest job seen so far. These lower bounds on the optimum makespan are immediate but have not been used in earlier strategies.

Formally, for $j = 1, \ldots, m$, let L_j^t be the *average load* of the $m - j + 1$ least loaded machines $M_j^t, \ldots, M_m^t$, i.e. $L_j^t = \frac{1}{m-j+1} \sum_{r=j}^{m} l_r^t$. We let $L^t = L_1^t = \frac{1}{m} \sum_{s=1}^{t} p_s$ be the average load of all the machines. For any $j = 1, \ldots, n$, let P_j^t be the processing time of the j-th largest job among the first t jobs $J_{\sigma(1)}, \ldots, J_{\sigma(t)}$ in $\mathcal{J}^\sigma$. If $t < j$, we set $P_j^t = 0$. We let $p_{\max}^t = P_1^t$ be the processing time of the largest job among the first t jobs in $\mathcal{J}^\sigma$. Finally, let $L = L^n$, $P_j = P_j^n$ and $p_{\max} = p_{\max}^n$.

The value $O^t = \max\{L^t, p_{\max}^t, 2P_{m+1}^t\}$ is a common lower bound on the optimum makespan for the first t jobs and hence $OPT(\mathcal{J})$, see Proposition 5 in the next section. Note that immediately before $J_{\sigma(t)}$ is scheduled, ALG can compute L^t and hence O^t because L^t is $1/m$ times the total processing time of the jobs that have arrived so far.

We next characterize load imbalance. Let

$$k = 2i - m \approx (4c - 7)m \approx 0.3912m$$

and

$$\alpha = \frac{2(c-1)}{2c-3} \approx 2.7376.$$

The *schedule at time* t is the one immediately before $J_{\sigma(t)}$ has to be assigned. The schedule is *flat* if $l_k^{t-1} < \alpha L_{i+1}^{t-1}$. Otherwise it is *steep*. Job $J_{\sigma(t)}$ is scheduled *flatly* (*steeply*) if the schedule at time t is flat (steep).

ALG handles each incoming job $J_{\sigma(t)}$, with processing time $p_{\sigma(t)}$, as follows. If the schedule at time t is steep, the job is placed on the least loaded machine M_m^{t-1}. On the other hand, if the schedule is flat, the machines M_i^{t-1}, M_{m-h}^{t-1} and M_m^{t-1} are probed in this order. If $l_i^{t-1} + p_{\sigma(t)} \leq c \cdot O^t$, then the new machine load on M_i^{t-1} will not violate the desired competitiveness. The job is placed on this machine M_i^{t-1}. Otherwise, if the latter inequality is violated, *ALG* checks if a placement on M_{m-h}^{t-1} is safe, i.e. if $l_{m-h}^{t-1} + p_{\sigma(t)} \leq c \cdot O^t$. If this is the case, the job is put on M_{m-h}^{t-1}. Otherwise, $J_{\sigma(t)}$ is finally scheduled on the least loaded machine M_m^{t-1}. A pseudo-code description of *ALG* is given below. The job assignment rules are also illustrated in Figures 1 and 2.

Algorithm 1 The scheduling algorithm *ALG*.

Let $J_{\sigma(t)}$ be the next job to be scheduled.
if the schedule at time t is steep **then**
    Assign $J_{\sigma(t)}$ to the least loaded machine M_m^{t-1};
else // the schedule is flat
    if $l_i^{t-1} + p_{\sigma(t)} \leq c \cdot O^t$ **then** Assign $J_{\sigma(t)}$ to M_i^{t-1};
    else if $l_{m-h}^{t-1} + p_{\sigma(t)} \leq c \cdot O^t$ **then** Assign $J_{\sigma(t)}$ to M_{m-h}^{t-1};
    else Assign $J_{\sigma(t)}$ to M_m^{t-1};

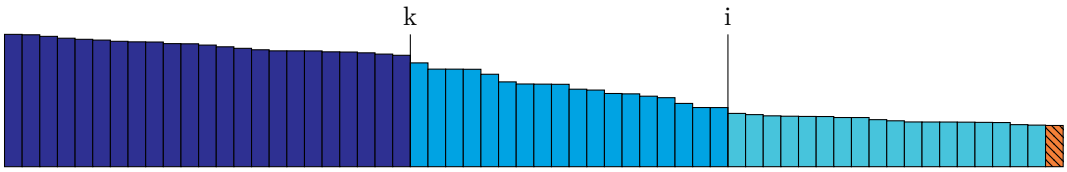

Figure 1 A steep schedule. *ALG* only considers the least loaded machine.

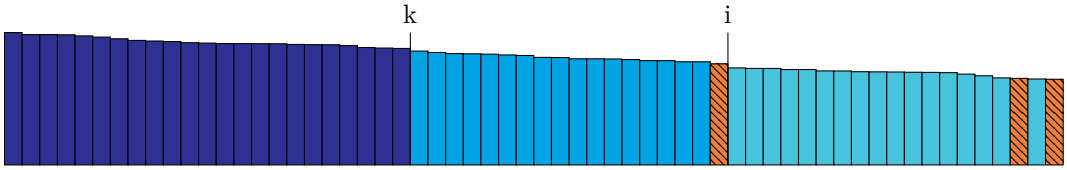

Figure 2 A flat schedule. The three machines considered by *ALG* are marked for $h = 2$.

In the next section we will prove the following theorem, Theorem 3, which uses the notion from Section 2. Lemma 2 then immediately gives the main result, Corollary 4.

▶ **Theorem 3.** *ALG is nearly c-competitive, with* $c < 1.8478$ *defined as above.*

▶ **Corollary 4.** *ALG is c-competitive in the random-order model as* $m \to \infty$.

From our analysis it can be verified that the number of machines required to be $(c + \varepsilon)$-competitive is bounded by a small polynomial of degree 4 in $1/\varepsilon$. For ease of presentation, we made no optimizations in that regard.

4 Analysis of the algorithm

4.1 Analysis basics

We present some results for the adversarial model so that we can focus on the true random-order analysis of *ALG* in the next sections. First, recall the three common lower bounds used for online makespan minimization.

▶ **Proposition 5.** *For any $\mathcal{J}$, there holds $OPT(\mathcal{J}) \geq \max\{L, p_{\max}, 2P_{m+1}\}$. Moreover, for any permutation J^σ, there holds $O^1 \leq O^2 \leq \ldots \leq O^n \leq OPT(\mathcal{J})$.*

Proof. The optimum makespan $OPT(\mathcal{J})$ cannot be smaller than the average machine load L for the input, even if all the jobs are distributed evenly among the m machines. Moreover, the job with the largest processing time $p_{\max}$ must be scheduled non-preemptively on one of the machines in an optimal schedule. Thus $OPT(\mathcal{J}) \geq p_{\max}$. Finally, among the $m+1$ largest jobs of the input, two must be placed on the same machine in an optimal solution. Hence $OPT(\mathcal{J}) \geq 2P_{m+1}$. For any permutation J^σ, the value O^t cannot decrease as jobs J_t arrive. ◀

For any job sequence $\mathcal{J} = J_1, \ldots, J_n$, let $R(\mathcal{J}) = \min\{\frac{L}{p_{\max}}, \frac{p_{\max}}{L}\}$. Intuitively, this measures the complexity of $\mathcal{J}$.

▶ **Proposition 6.** *For any $\mathcal{J} = J_1, \ldots, J_n$, there holds $ALG(\mathcal{J}) \leq \max\{1+R(\mathcal{J}), c\}OPT(\mathcal{J})$.*

Proof. Let $\mathcal{J} = J_1, \ldots, J_n$ be an arbitrary job sequence and let J_t be the job that defines ALG's makespan. If the makespan exceeds $c \cdot OPT(\mathcal{J})$, then it exceeds $c \cdot O^t$. Thus ALG placed J_t on machine M_m^{t-1}, cf. lines 4 and 5 of the algorithm. This machine was a least loaded one, having a load of at most L. Hence $ALG(\mathcal{J}) \leq L + p_t \leq L + p_{\max} \leq \frac{L+p_{\max}}{\max\{L,p_{\max}\}} \cdot OPT(\mathcal{J}) = (1 + R(\mathcal{J})) \cdot OPT(\mathcal{J})$. ◀

Since $R(\mathcal{J}) \leq 1$ we immediately obtain the following result, which ensures that ALG satisfies the first condition of a nearly c-competitive algorithm, see Definition 1.

▶ **Corollary 7.** *ALG is 2-competitive in the adversarial model.*

We next identify a class of *plain* job sequences that we do not need to consider in the random-order analysis because ALG's makespan is upper bounded by c times the optimum on these inputs.

▶ **Definition 8.** *A job sequence $\mathcal{J} = J_1, \ldots, J_n$ is called* plain *if $n \leq m$ or if $R(\mathcal{J}) \leq c - 1$. Otherwise it is called* proper*.*

Let $\mathcal{J} = J_1, \ldots, J_n$ be any job sequence that is processed/scheduled in this order. Observe that if it contains at most m jobs, i.e. $n \leq m$, and ALG cannot place a job J_t on machines M_i^{t-1} or M_{m-h}^{t-1} because the resulting load would exceed $c \cdot O^t$, then the job is placed on an empty machine. Using Proposition 6 we derive the following fact.

▶ **Lemma 9.** *For any plain job sequence $\mathcal{J} = J_1, \ldots, J_n$, there holds $ALG(\mathcal{J}) \leq c \cdot OPT(\mathcal{J})$.*

If a job sequence $\mathcal{J}$ is plain (proper), then every permutation of it is. Hence, given Lemma 9, we may concentrate on proper job sequences in the remainder of the analysis. We finally state a fact that relates to the second condition of a nearly c-competitive algorithm, see again Definition 1. The proof is given in the full version.

▶ **Lemma 10.** *Let $\mathcal{J} = J_1, \ldots, J_n$ be any job sequence that is scheduled in this order and let J_t be a job that causes ALG's makespan to exceed $(c+\varepsilon)OPT(\mathcal{J})$, for some $\epsilon \geq 0$. Then both the load of ALG's least loaded machine at the time of the assignment as well as p_t exceed $(c-1+\varepsilon)OPT(\mathcal{J})$.*

Proof. ALG places J_t on machine M_m^{t-1}, which is a least loaded machine when the assignment is done. If l_m^{t-1} or p_t were upper bounded by $(c-1+\varepsilon)OPT(\mathcal{J})$, then the resulting load would be $l_m^{t-1} + p_t \leq (c-1+\varepsilon)OPT(\mathcal{J}) + \max\{L, p_t\} \leq (c-1+\varepsilon)OPT(\mathcal{J}) + OPT(\mathcal{J}) = (c+\varepsilon)OPT(\mathcal{J})$. ◀

4.2 Stable job sequences

We define the class of stable job sequences. These sequences are robust in that they will admit an adversarial analysis of ALG. Intuitively, the sequences reveal information on the largest jobs when a significant fraction of the total processing volume $\sum_{t=1}^{n} p_t$ has been scheduled. More precisely, one gets an estimate on the processing time of the h-th largest job in the entire sequence and encounters a relevant number of the $m+1$ largest jobs. If a job sequence is unstable, large jobs occur towards the very end of the sequence and can cause a high makespan relative to the optimum one.

We will show that ALG is adversarially $(c+\varepsilon)$-competitive on stable sequences, for a given $\varepsilon > 0$. Therefore, the definition of stable sequences is formulated for a fixed $\varepsilon > 0$. Given $\mathcal{J}$, let $\mathcal{J}^\sigma = J_{\sigma(1)}, \ldots, J_{\sigma(n)}$ be any permutation of the jobs. Furthermore, for every $j \leq n$ and in particular $j \in \{h, m+1\}$, the *set of the j largest jobs* is a fixed set of cardinality j such that no job outside this set has a strictly larger processing time than any job inside the set.

▶ **Definition 11.** *A job sequence $\mathcal{J}^\sigma = J_{\sigma(1)}, \ldots, J_{\sigma(n)}$ is* stable *if the following conditions hold.*

- *There holds $n > m$.*
- *Once $L^t \geq (c-1)\frac{i}{m}L$, there holds $p^t_{\max} \geq P_h$.*
- *For every $j \geq i$, the sequence ending once we have $L^t \geq (\frac{j}{m} + \frac{\varepsilon}{2})L$ contains at least $j+h+2$ many of the $m+1$ largest jobs in $\mathcal{J}$.*
- *The sequence ending right before either (a) $L^t \geq \frac{i}{m}(c-1)\varepsilon L$ holds or (b) the h-th largest job of $\mathcal{J}$ is scheduled contains at least $h+1$ many of the $m+1$ largest jobs in $\mathcal{J}$.*

Otherwise the job sequence is unstable.

Given $\varepsilon > 0$ and m, let $P_\varepsilon(m)$ be the infimum, over all proper job sequences $\mathcal{J}$, that a random permutation of $\mathcal{J}$ is stable, i.e.

$$P_\varepsilon(m) = \inf_{\mathcal{J} \text{ proper}} \mathbf{P}_{\sigma \sim S_n}[\mathcal{J}^\sigma \text{ is stable}].$$

As the main result of this section we will prove that this probability tends to 1 as $m \to \infty$.

▶ **Main Lemma 1.** *For every $\varepsilon > 0$, there holds $\lim_{m\to\infty} P_\varepsilon(m) = 1$.*

The above lemma implies that for any $\varepsilon > 0$ there exists an $m(\varepsilon)$ such that, for all $m \geq m(\varepsilon)$ and all $\mathcal{J}$, there holds $\mathbf{P}_{\sigma\sim S_n}[\mathcal{J}^\sigma \text{ is stable}] \geq 1-\varepsilon$. In Section 4.3 we will show that ALG is $(c+\varepsilon)$-competitive on stable job sequences. This implies $\mathbf{P}_{\sigma\sim S_n}[ALG(\mathcal{J}^\sigma) \geq (c+\varepsilon)OPT(\mathcal{J})] \leq \varepsilon$. Given Lemma 7, we obtain the following corollary.

▶ **Corollary 12.** *If ALG is adversarially $(c+\varepsilon)$-competitive on stable sequences, for every $\varepsilon > 0$ and $m \geq m(\varepsilon)$ sufficiently large, then it is nearly c-competitive.*

In the remainder of this section we describe how to establish Main Lemma 1. Full proofs of all the lemmas of this section are given in the full version. We need some notation. In Section 3 the value L^t_j was defined with respect to a fixed job sequence that was clear from the context. We adopt the notation $L^t_j[\mathcal{J}^\sigma]$ to make this dependence visible. We adopt a similar notation for the variables L, P^t_j, P_j, $p^t_{\max}$ and $p_{\max}$. For an input $\mathcal{J}$ and $\sigma \in S_n$, we will use the notation $L^t_j[\sigma] = L^t_j[\mathcal{J}^\sigma]$. Again, we use a similar notation for the variables P^t_j and $p^t_{\max}$.

At the heart of the proof of Main Lemma 1 is the Load Lemma. Observe that after t time steps in a random permutation of an input $\mathcal{J}$, each job has arrived with probability t/n. Thus the expected total processing time of the jobs seen so far is $t/n \cdot \sum_{s=1}^{n} p_s$. Equivalently, in expectation L^t equals $t/n \cdot L$. The Load Lemma proves that this relation holds with high probability. We set $t = \varphi n$.

▶ **Load Lemma.** *Given any $\varepsilon > 0$ and $\varphi \in (0, 1]$, there exists an $m(\varepsilon, \varphi)$ such that for all $m \geq m(\varepsilon, \varphi)$ and all proper sequences $\mathcal{J}$, there holds*

$$\mathbf{P}_{\sigma \sim S_n}\left[\left|\frac{L^{\lfloor \varphi n \rfloor}[\mathcal{J}^\sigma]}{\varphi L[\mathcal{J}^\sigma]} - 1\right| \geq \varepsilon\right] \leq \varepsilon.$$

Proof sketch. By scaling all job processing times by a common factor we may assume that $p_{\max} = 1$. Then $L = \Theta(m)$ because $\mathcal{J}$ is proper. The main idea of the proof is to show that the variance of the random variable $L^{\lfloor \varphi n \rfloor}[\mathcal{J}^\sigma]$ lies in $O(m) = O(L)$. Using Chebyshev's inequality we show that the probability of $L^{\lfloor \varphi n \rfloor}[\mathcal{J}^\sigma]$ deviating by its expected value φL by more than some term in $\Theta(m^{-1/4})$ is in $O(m^{-1/2})$. The lemma then follows by choosing m sufficiently large. ◀

We note that the Load Lemma does not hold for general sequences. A counterexample is a job sequence in which one job carries all the load, while all the other jobs have a negligible processing time. The proof of the Load Lemma relies on a lower bound of $R(\mathcal{J})$, which is $c - 1$ for proper sequences.

We present two consequences of the Load Lemma that will allow us to prove that stable sequences reveal information on the largest jobs when a certain processing volume has been scheduled. Consider a proper $\mathcal{J}$. Given $\mathcal{J}^\sigma = J_{\sigma(1)}, \ldots, J_{\sigma(n)}$ and $\varphi > 0$, let $N(\varphi)[\mathcal{J}^\sigma]$ be the number of jobs $J_{\sigma(t)}$ that are among the $m + 1$ largest jobs in $\mathcal{J}$ and such that $L^t \leq \varphi L$.

▶ **Lemma 13.** *Let $\varepsilon > 0$ and $\varphi \in (0, 1]$. Then there holds*

$$\lim_{m \to \infty} \inf_{\mathcal{J} \text{ proper}} \mathbf{P}_{\sigma \sim S_n}\left[N(\varphi + \varepsilon)[\mathcal{J}^\sigma] \geq \lfloor \varphi m \rfloor + h + 2\right] = 1.$$

Proof sketch. The Load Lemma basically matches load ratios L^t/L with ratios t/n on the time line of job arrivals, up to some margin of error. We can then infer that at least $\lfloor \varphi m \rfloor + h + 1$ of the $m + 1$ largest jobs are among the first $(\varphi + \varepsilon)n$ jobs in a job sequence $\mathcal{J}^\sigma$, with a probability that tends to 1 as $m \to \infty$. In expectation $(\varphi + \varepsilon)(m + 1)$ of the $m + 1$ largest jobs occur in this prefix, which is strictly more than $\lfloor \varphi m \rfloor + h + 1$, for m large enough. Formally, we show that (a slight variant of) the random variable $N(\varphi + \varepsilon)[\mathcal{J}^\sigma]$ is hypergeometrically distributed and has variance at most $m + 1$. Using Chebyshev's inequality we derive Lemma 13. ◀

▶ **Lemma 14.** *Let $\varepsilon > 0$ and $\varphi \in (0, 1]$. Then there holds*

$$\lim_{m \to \infty} \inf_{\mathcal{J} \text{ proper}} \mathbf{P}_{\sigma \sim S_n}\left[\forall_{\tilde{\varphi} \geq \varphi}\ N(\tilde{\varphi} + \varepsilon)[\mathcal{J}^\sigma] \geq \lfloor \tilde{\varphi} m \rfloor + h + 2\right] = 1.$$

Proof sketch. By rounding the values $\tilde{\varphi}$ we may restrict ourselves to finitely many $\tilde{\varphi}$. Using the Union Bound and Lemma 13 we can prove Lemma 14. ◀

Proof sketch for Main Lemma 1. We consider the properties in the definition of stable job sequences. Since $\mathcal{J}$ is proper, there holds $n > m$. By the Load Lemma the second property translates to one of the h largest jobs being among the first $(c - 1)\frac{i}{m}n$ jobs in the permuted sequence $\mathcal{J}^\sigma$. The corresponding probability is roughly $1 - (1 - (c - 1)\frac{i}{m})^h$ and (quickly)

approaches 1 as m and thus h tends to infinity. The third property is a consequence of Lemma 14. For the fourth property we use Lemma 13, considering the sequence ending once $L^t \geq (\frac{j}{m} + \frac{\varepsilon}{2})L$ holds. Finally, the probability of the h-th largest job being preceded by at least $h+1$ of the $m+1$ largest jobs approaches 1 since $h \in o(\sqrt{m})$. Again a full proof is given in the full version. ◀

4.3 An adversarial analysis

In this section we prove the following main result.

▶ **Main Lemma 2.** *For every $\varepsilon > 0$ and $m \geq m(\varepsilon)$ sufficiently large, ALG is adversarially $(c+\varepsilon)$-competitive on stable job sequences.*

Consider a fixed $\varepsilon > 0$. Given Lemma 7, we may assume that $0 < \varepsilon < 2 - c$. Suppose that there was a stable job sequence $\mathcal{J}^\sigma$ such that $ALG(\mathcal{J}^\sigma) > (c+\varepsilon)OPT(\mathcal{J}^\sigma)$. We will derive a contradiction, given that m is large. In order to simplify notation, in the following let $\mathcal{J} = \mathcal{J}^\sigma$ be the stable job sequence violating the performance ratio of $c+\varepsilon$. Let $\mathcal{J} = J_1, \ldots, J_n$ and $OPT = OPT(\mathcal{J})$.

Let $J_{n'}$ be the first job that causes ALG to have a makespan greater than $(c+\varepsilon)OPT$ and let $b_0 = l_m^{n'-1}$ be the load of the least loaded machine $M_m^{n'-1}$ right before $J_{n'}$ is scheduled on it. The makespan after $J_{n'}$ is scheduled, called the *critical makespan*, is at most $b_0 + p_{n'} \leq b_0 + OPT$. In particular $b_0 > (c-1+\varepsilon)OPT$ as well as $p_{n'} > (c-1+\varepsilon)OPT$, see Lemma 10. Let

$$\lambda_{\text{start}} = \tfrac{c-1}{1+2c(2-c)} \approx 0.5426 \quad \text{and} \quad \lambda_{\text{end}} = \tfrac{1}{2(c-1+\varepsilon)} \approx 0.5898.$$

There holds $\lambda_{\text{start}} < \lambda_{\text{end}}$. The critical makespan of ALG is bounded by $b_0 + OPT < (1 + \frac{1}{c-1+\varepsilon})b_0 = (c+\varepsilon)\frac{b_0}{c-1+\varepsilon} = (c+\varepsilon)2\lambda_{\text{end}}b_0$. Since ALG does not achieve a performance ratio of $c+\varepsilon$ on $\mathcal{J}$ we have

$$P_{m+1} \leq OPT/2 < \lambda_{\text{end}}b_0. \tag{1}$$

Our main goal is to derive a contradiction to this inequality.

The impact of the variable P_h

A new, crucial aspect in the analysis of ALG is P_h, the processing time of the h-th largest job in the sequence $\mathcal{J}$. Initially, when the processing of $\mathcal{J}$ starts, we have no information on P_h and can only infer $P_{m+1} \geq \lambda_{\text{start}}b_0$. The second property in the definition of stable job sequences ensures that $p_{\max}^t \geq P_h$ once the load ratio L^t/L is sufficiently large. Note that ALG then also works with this estimate because $P_h \leq p_{\max}^t \leq O^t$. This will allow us to evaluate the processing time of flatly scheduled jobs. In order prove that P_{m+1} is large, we will relate P_{m+1} and P_h, i.e. we will lower bound P_{m+1} in terms of P_h and vice versa. Using the relation we can then conclude $P_{m+1} \geq \lambda_{\text{end}}b_0$. In the analysis we repeatedly use the properties of stable job sequences and will explicitly point to it when this is the case. The omitted proofs of propositions and lemmas are given in the full version of the paper.

We next make the relationship between P_h and P_{m+1} precise. Given $0 < \lambda$, let $f(\lambda) = 2c\lambda - 1$ and given $w > 0$, let $g(w) = (c(2c-3)-1)w + 4 - 2c \approx 0.2854 \cdot w + 0.3044$. We set $g_b(\lambda) = g\left(\frac{\lambda}{b}\right)b$ and $f_b(w) = f\left(\frac{w}{b}\right)b$, for any $b > 0$. Then we will lower bound P_{m+1} by $g_{b_0}(P_h)$ and P_h by $f_{b_0}(P_{m+1})$. We state two technical propositions.

▶ **Proposition 15.** *For $\lambda > \lambda_{\text{start}}$, we have $g(f(\lambda)) > \lambda$.*

▶ **Proposition 16.** *For $0 < \varepsilon \leq 1$, we have $g(1-\varepsilon) > \lambda_{\text{end}}$.*

We leave the proof of Proposition 15 to the full version of the paper. Proposition 16 determines the choice of our competitive ratio c. Recall that c is chosen minimal such that $Q[c] = 4c^3 - 14c^2 + 16c - 7 \geq 0$.

Proof of Proposition 16. We calculate that

$$\begin{aligned} g(1-\varepsilon) - \lambda_{\text{end}} &= (c(2c-3)-1)(1-\varepsilon) + 4 - 2c - \frac{1}{2(c-1+\varepsilon)} \\ &= \frac{2(c-1+\varepsilon)(2c^2-5c+3-(2c^2-3c-1)\varepsilon) - 1}{2(c-1+\varepsilon)} \\ &= \frac{4c^3-14c^2+16c-7+(4-2c)\varepsilon - 2(2c^2-3c-1)\varepsilon^2}{2(c-1+\varepsilon)}. \end{aligned}$$

Recall that $Q[c] = 4c^3 - 14c^2 + 16c - 7 = 0$. For $0 < \varepsilon \leq 1$ we have

$$(4-2c)\varepsilon - (2c^2-3c-1)\varepsilon^2 \approx 0.3044 \cdot \varepsilon - 0.2854 \cdot \varepsilon^2 > 0.$$

Thus we see that $g(1-\varepsilon) - \lambda_{\text{end}} > 0$ and can conclude the lemma. ◀

4.3.1 Analyzing large jobs towards lower bounding P_h and P_{m+1}

Let $b > (c-1+\varepsilon)OPT$ be a value such that immediately before $J_{n'}$ is scheduled at least $m-h$ machines have a load of at least b. Note that $b = b_0$ satisfies this condition but we will be interested in larger values of b as well. We call a machine *b-full* once its load is at least b; we call a job J a *b-filling job* if it causes the machine it is scheduled on to become b-full. We number the b-filling jobs according to their order of arrival $J^{(1)}, J^{(2)}, \ldots$ and let $t(j)$ denote the time of arrival of the j-th filling job $J^{(j)}$.

Recall that our main goal is to show that $P_{m+1} \geq \lambda_{\text{end}} b_0$ holds. To this end we will prove that the *b_0-filling jobs* have a processing time of at least $\lambda_{\text{end}} b_0$. As there are m such jobs, the bound on P_{m+1} follows by observing that $J_{n'}$ arrives after all *b_0-filling jobs* are scheduled and that its processing time exceeds $\lambda_{\text{end}} b_0$ as well. In fact, since $OPT \geq b_0$, we have

$$p_{n'} > (c-1)OPT > 0.847 \cdot OPT > \lambda_{\text{end}} b_0 \approx 0.5898 \cdot b_0. \tag{2}$$

We remark that different to previous analyses in the literature we do not solely rely on lower bounding the processing time of filling jobs. By using the third property of stable job sequences, we can relate load and the size of the $(m+1)$-st largest job at specific points in the time horizon, cf. Lemma 22.

In the following we regard b as fixed and omit it from the terms *filling job* and *full*. Let $\lambda = \max\{\lambda_{\text{start}} b, \min\{g_b(P_h), \lambda_{\text{end}} b\}\}$. We call a job *large* if it has a processing time of at least λ. Let $\tilde{t} = t(m-h)$ be the time when the $(m-h)$-th filling job arrived. The remainder of this section is devoted to showing the following important Lemma 17. Some of the underlying lemmas, but not all of them, hold if $m \geq m(\varepsilon)$ is sufficiently large. We will make the dependence clear.

▶ **Lemma 17.** *At least one of the following statements holds:*

- *All filling jobs are large.*
- *If $m \geq m(\varepsilon)$, there holds $P^{\tilde{t}}_{m+1} \geq \lambda = \max\{\lambda_{\text{start}} b, \min\{g_b(P_h), \lambda_{\text{end}} b\}\}$, i.e. there are at least $m+1$ large jobs once the $(m-h)$-th filling job is scheduled.*

Before we prove the lemma we derive two important implications towards a lower bound of P_{m+1}.

▶ **Lemma 18.** *We have* $P_{m+1} \geq \lambda = \max\{\lambda_{\text{start}} b_0, \min\{g_{b_0}(P_h), \lambda_{\text{end}} b_0\}\}$.

Proof. Apply the last lemma, taking into account that $b \geq b_0$, and use that there are m many b_0-filling jobs followed by $J_{n'}$. The latter has size at least λ by inequality (2). ◀

We also want to lower bound the processing time of the $(m+1)$-st largest job at time $\tilde{t}$. However, at that time only $m-h$ filling jobs have arrived. The next lemma ensures that, if additionally P_h is not too large, this is not a problem.

▶ **Lemma 19.** *If* $P_h \leq (1-\varepsilon)b$ *and* $m \geq m(\varepsilon)$*, the second statement in Lemma 17 holds, i.e.* $P^{\tilde{t}}_{m+1} \geq \lambda = \max\{\lambda_{\text{start}} b, \min\{g_b(P_h), \lambda_{\text{end}} b\}\}$.

The proof of the lemma makes use of the fourth property of stable job sequences.

We introduce late and early filling jobs. We need a certain condition to hold, see Lemma 22, in order to show that the early filling jobs are large. We show that if this condition is not met, the fact that the given job sequence is stable ensures that $P^{\tilde{t}}_m \geq \lambda$.

Let s be chosen maximal such that the s-th filling job is scheduled steeply. If $s \leq i$, then set $s = i+1$ instead. We call all filling jobs $J^{(j)}$ with $j > i$ that are scheduled flatly *late filling jobs*. All other filling jobs are called *early filling jobs*. In particular the job $J^{(s+1)}$ and the filling jobs afterwards are late filling jobs. The following proposition implies that the fillings jobs after $J^{(m-h)}$, if they exist, are all late, i.e. scheduled flatly.

▶ **Proposition 20.** *We have* $s \leq m-h$ *if* $m \geq m(\varepsilon)$.

We need a technical lemma. For any time t, let $\overline{L}^t_s = \frac{1}{m-h-s+1} \sum_{j=s}^{m-h} l^t_j$ be the average load on the machines numbered s to $m-h$.

▶ **Lemma 21.** *If* $\overline{L}^{t(s)-1}_s \geq \alpha^{-1} b$ *holds and* $m \geq m(\varepsilon)$*, we have* $L^{t(s)-1} > \left(\frac{s}{m} + \frac{\varepsilon}{2}\right) \cdot L$.

▶ **Lemma 22.** *If the late filling jobs are large,* $\overline{L}^{t(s)-1}_s \geq \alpha^{-1} b$ *and* $m \geq m(\varepsilon)$*, we have* $P^{\tilde{t}}_{m+1} \geq \lambda$.

Proof. Assume that the conditions of the lemma hold. By Lemma 21 we have $L^{t(s)-1} > \left(\frac{s}{m} + \frac{\varepsilon}{2}\right) \cdot L$. By the third property of stable sequences, at most $m+1-(s+h+2) = m-s-h-1$ of the largest $m+1$ jobs appear in the sequence starting after time $t(s)-1$. However, this sequence contains $m-h-s$ late filling jobs. Thus there exists a late filling job that is not among the $m+1$ largest jobs. As it has a processing time of at least λ, by the assumption of the lemma, $P_{m+1} \geq \lambda$ holds.

Now consider the $m+1$ largest jobs of the entire sequence that arrive before $J^{(s)}$ as well as the jobs $J^{(s+1)}, \ldots, J^{(m-h)}$. There are at least $s+h+2$ of the former and $m-h-s$ of the latter. Thus we have found a set of at least $m+1$ jobs arriving before (or at) time $\tilde{t} = t(m-h)$. Moreover, we argued that all these jobs have a processing time of at least λ. Hence $P^{\tilde{t}}_{m+1} \geq \lambda$ holds true. ◀

We are ready to evaluate the processing time of filling jobs to prove Lemma 17.

▶ **Lemma 23.** *The processing time of late filling jobs strictly exceeds* $\max\{\lambda_{\text{start}} b, g_b(P_h)\}$.

▶ **Lemma 24.** *If* $\overline{L}^{t(s)-1}_s < \alpha^{-1} b$ *holds, the early filling jobs have a processing time of at least* $\lambda_{\text{end}} b$.

Before proving Lemma 24 let us observe the following, strengthening its condition.

▶ **Lemma 25.** *We have*

$$L_{i+1}^{t(i+1)-1} \leq L_{i+2}^{t(i+2)-1} \leq \dots L_s^{t(s)-1}.$$

Proof. Let $i+1 \leq j < s$. It suffices to verify that

$$L_j^{t(j)-1} \leq L_{j+1}^{t(j)} \leq L_{j+1}^{t(j+1)-1}.$$

The second inequality is obvious because for every r the loads l_r^t can only increase as t increases. For the first inequality we note that by definition the job $J^{(j)}$ was scheduled steeply and hence on a least loaded machine. This machine became full. Thus it is not among the $m-j$ least loaded machines at time $t(j)$. In particular $L_{j+1}^{t(j)}$, the average over the $m-j$ smallest loads at time $t(j)$, is also the average of the $m-j+1$ smallest loads excluding the smallest load at time $t(j)-1$. Therefore it cannot be less than $L_j^{t(j)-1}$. ◀

Proof of Lemma 24. Let $i < j \leq s$ such that $J^{(j)}$ was an early filling job. By Lemma 25 we have $L_j^{t(j)-1} \leq L_s^{t(s)-1} < \alpha^{-1}b = b - \frac{b}{2(c-1)} < b - \lambda_{\text{end}}b$. By definition $J^{(j)}$ was scheduled on a least loaded machine $M_m^{t(j)-1}$ which had load less than $L_j^{t(j)-1} < b - \lambda_{\text{end}}b$ before and at least b afterwards because it became full. In particular $J^{(j)}$ had size $\lambda_{\text{end}}b$.

For $k < j \leq i$ the job $J^{(j)}$ is scheduled steeply because we have by Lemma 25

$$l_k^{t(j)-1} \geq b > \alpha L_s^{t(s)-1} \geq \alpha L_{i+1}^{t(i+1)-1} \geq \alpha L_{i+1}^{t(j)-1}.$$

Thus for $k < j \leq i$ the job $J^{(j)}$ is scheduled on the least loaded machine $M_m^{t(j)-1}$, whose load $l_m^{t(j)-1}$ is bounded by

$$l_m^{t(j)-1} \leq L_{i+1}^{t(j)-1} \leq L_s^{t(s)-1} < \alpha^{-1}b = b - \frac{b}{2(c-1)} < b - \lambda_{\text{end}}b.$$

Hence the job $J^{(j)}$ had a size of at least $\lambda_{\text{end}}b$. We also observe that we have

$$l_i^{t(k)-1} \leq l_{i+1}^{t(k+1)-1} \leq \dots \leq l_{i+(m-i)}^{t(k+(m-i))-1} = l_m^{t(i)-1} < b - \lambda_{\text{end}}b.$$

In particular for $1 \leq j \leq k$ any filling job $J^{(j)}$ filled a machine with a load of at most $\max\{l_m^{t(k)}, l_i^{t(k)}\} = l_i^{t(k)} < b - \lambda_{\text{end}}b$. Hence it had a size of at least $\lambda_{\text{end}}b$. ◀

We now conclude the main lemma of this subsection, Lemma 17.

Proof of Lemma 17. By Lemma 23, all late filling jobs are large. We distinguish two cases depending on whether or not $\overline{L}_s^{t(s)-1} < \alpha^{-1}b$ holds. If it does, all filling jobs are large by Lemma 24 and the first statement in Lemma 17 holds. Otherwise, the second statement in Lemma 17 holds by Lemma 22. ◀

4.3.2 Lower bounding P_h and P_{m+1}

In this section we establish the following relations on P_h and P_{m+1}.

▶ **Lemma 26.** *There holds $P_h > (1-\varepsilon)b_0$ or $P_{m+1} \geq \lambda_{\text{end}}b_0$ if $m \geq m(\varepsilon)$.*

For the proof we need a way to lower bound the processing time of a job J_t depending on P_{m+1}^t:

▶ **Lemma 27.** *Let J_t be any job scheduled flatly on the least loaded machine and let $b = l_{m-h}^{t-1}$ be the load of the $(h+1)$-th least loaded machine. Then J_t has a processing time of at least $f_b(P_{m+1}^t)$.*

Proof. From the fact that J_t was not scheduled on the $(h+1)$-th least loaded machine M^t_{m-h} we derive that $p_t > c \cdot O^t - b \geq c \cdot P^t_{m+1} - b = f_b(P^t_{m+1})$ holds. ◀

Proof of Lemma 26. Assume for a contradiction that we had $P_h \leq (1-\varepsilon)b_0$. Let $J = J_t$ be the smallest among the h last b_0-filling jobs. Then J has a processing time $p \leq P_h$. We want to derive a contradiction to that. Let $b_1 = l^{t-1}_{m-h}$ be the load of the $(m-h)$-th machine right before J was scheduled. Because this machine was b_0-full at that time we know that $b_1 \geq b_0 > (c-1+\varepsilon)OPT$ holds and it makes sense to consider b_1-filling jobs. Let $\tilde{t}$ be the time the $(m-h)$-th b_1-filling job arrived. By Lemma 17 we have $P^{\tilde{t}}_{m+1} \geq \lambda = \max\{\lambda_{\text{start}} b_1, \min\{g_{b_1}(P_h), \lambda_{\text{end}} b_1\}\}$.

If we have $\lambda = \lambda_{\text{end}} b_1 \geq \lambda_{\text{end}} b_0$ we have already proven $P_{m+1} \geq \lambda_{\text{end}} b_0$ and the lemma follows. So we are left to treat the case that we have $P^{\tilde{t}}_{m+1} \geq \lambda = \max\{\lambda_{\text{start}} b_1, g_{b_1}(P_h)\}$.

Now we can derive the following contradiction:

$$P^{\tilde{t}}_{m+1} \geq g_{b_1}(P_h) \geq g_{b_1}(p) \geq g_{b_1}\left(f_{b_1}\left(P^{\tilde{t}}_{m+1}\right)\right) = g\left(f\left(\frac{P^{\tilde{t}}_{m+1}}{b_1}\right)\right) b_1 > P^{\tilde{t}}_{m+1}.$$

For the second inequality, we use the monotonicity of $g_{b_1}(-)$. The third inequality follows from Lemma 27 and the last one from Proposition 15. ◀

4.3.3 Establishing Main Lemma 2

Let $m \geq m(\varepsilon)$ be sufficiently large. The machine number $m(\varepsilon)$ is determined by the proofs of Proposition 20 and Lemma 21, and then carries over to the subsequent lemmas. Let us assume for a contradiction sake that there was a stable sequence $\mathcal{J}$ such that $ALG(\mathcal{J}) > (c+\varepsilon)OPT(\mathcal{J})$. As argued in the beginning of Section 4.3, see (1), it suffices to show that $P_{m+1} \geq \lambda_{\text{end}} b_0$. If this was not the case, we would have $P_h \geq (1-\varepsilon)b_0$ by Lemma 26. In particular by Proposition 16 we had $g_{b_0}(P_h) = g(1-\varepsilon)b_0 > \lambda_{\text{end}} b_0$. But now Lemma 18 shows that $P_{m+1} \geq \max\{\lambda_{\text{start}} b_0, \min\{g_{b_0}(P_h), \lambda_{\text{end}} b_0\}\} = \lambda_{\text{end}} b_0$.

We conclude, by Corollary 12, that ALG is nearly c-competitive.

5 Lower bounds

We present lower bounds on the competitive ratio of any deterministic online algorithm in the random-order model. Theorem 29 implies that if a deterministic online algorithm is c-competitive with high probability as $m \to \infty$, then $c \geq 3/2$.

▶ **Theorem 28.** *Let A be a deterministic online algorithm that is c-competitive in the random-order model. Then $c \geq 4/3$ if $m \geq 8$.*

▶ **Theorem 29.** *Let A be a deterministic online algorithm that is nearly c-competitive. Then $c \geq 3/2$.*

A basic family of inputs are job sequences that consist of jobs having an identical processing time of, say, 1. We first analyze them and then use the insight to derive our lower bounds. Let $m \geq 2$ be arbitrary. For any deterministic online algorithm A, let $r(A,m)$ be the maximum number in $\mathbb{N} \cup \{\infty\}$ such that A handles a sequence consisting of $r(A,m) \cdot m$ jobs with an identical processing time of 1 by scheduling each job on a least loaded machine.

▶ **Lemma 30.** *Let $m \geq 2$ be arbitrary. For every deterministic online algorithm A, there exists a job sequence $\mathcal{J}$ such that $A^{\text{rom}}(\mathcal{J}) \geq (1 + \frac{1}{r(A,m)+1})OPT(\mathcal{J})$. We use the convention that $\frac{1}{\infty+1} = 0$.*

Proof. For $r(A,m) = \infty$ there is nothing to show. For $r(A) < \infty$, consider the sequence $\mathcal{J}$ consisting of $(r(A,m)+1) \cdot m$ identical jobs, each having a processing time of 1. It suffices to analyze the algorithm adversarially as all permutations of the job sequence are identical. After having handled the first $r(A,m) \cdot m$ jobs, the algorithm A has a schedule in which every machine has load of $r(A,m)$. By the maximality of $r(A,m)$, the algorithm A schedules one of the following m jobs on a machine that is not a least loaded one. The resulting makespan is $r(A,m)+2$. The lemma follows since the optimal makespan is $r(A,m)+1$. ◀

Proof of Theorem 28. Let $m \geq 8$ be arbitrary. Consider any deterministic online algorithm A. If $r(A,m) \leq 2$, then, by Lemma 30, there exists a sequence $\mathcal{J}$ such that $A^{\text{rom}}(\mathcal{J}) \geq \frac{4}{3} \cdot OPT(\mathcal{J})$. Therefore, we may assume that $r(A,m) \geq 3$. Consider the input sequence $\mathcal{J}$ consisting of $4m-4$ identical small jobs of processing time 1 and one large job of processing time 4. Obviously $\text{OPT}(\mathcal{J}) = 4$.

Let i be the number of small jobs preceding the large job in $\mathcal{J}^\sigma$. The random variable i takes any (integer) value between 0 and $4m-4$ with probability $\frac{1}{4m-3}$. Since $r(A,m) \geq 3$ the least loaded machine has load of at least $l = \left\lfloor \frac{i}{m} \right\rfloor$ when the large job arrives. Thus $A(\mathcal{J}^\sigma) \geq l+4$. The load l takes the values 0, 1 and 2 with probability $\frac{m}{4m-3}$ and the value 3 with probability $\frac{m-3}{4m-3}$. Hence the expected makespan of algorithm A is at least

$$A^{\text{rom}}(\mathcal{J}) \geq \frac{m}{4m-3} \cdot (0+1+2) + \frac{m-3}{4m-3} \cdot 3 + 4 = \frac{6m-9}{4m-3} + 4 > \frac{16}{3} = \frac{4}{3}\text{OPT}(\mathcal{J}).$$

For the last inequality we use that $m \geq 8$. ◀

Proof of Theorem 29. Let $m \geq 2$ be arbitrary and let A be any deterministic online algorithm. If $r(A,m) = 0$, then consider the sequence $\mathcal{J}$ consisting of m jobs with a processing time of 1 each. On every permutation of $\mathcal{J}$ algorithm A has a makespan of 2, while the optimum makespan is 1. If $r(A,m) \geq 1$, then consider the sequence $\mathcal{J}$ consisting of $2m-2$ small jobs having a processing time of 1 and one large job with a processing time of 2. Obviously $OPT(\mathcal{J}) = 2$. If the permuted sequence starts with m small jobs, the least loaded machine has load 1 once the large job arrives. Under such permutations $A(\mathcal{J}^\sigma) \geq 3 = \frac{3}{2} \cdot \text{OPT}(\mathcal{J})$ holds true. The probability of this happening is $\frac{m-1}{2m-1}$. The probability approaches $\frac{1}{2}$ and in particular does not vanish, for $m \to \infty$. Thus, if A is nearly c-competitive, then $c \geq 3/2$. ◀

References

1 S. Albers. Better bounds for online scheduling. *SIAM Journal on Computing*, 29(2):459–473, 1999. Publisher: SIAM.

2 M. Babaioff, N. Immorlica, D. Kempe, and Robert Kleinberg. A knapsack secretary problem with applications. In *Proc. 10th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, pages 16–28. Springer, 2007.

3 M. Babaioff, N. Immorlica, David Kempe, and R. Kleinberg. Matroid Secretary Problems. *J. ACM*, 65(6):1–26, 2018. Publisher: ACM New York, NY, USA.

4 Y. Bartal, A. Fiat, H. Karloff, and R. Vohra. New algorithms for an ancient scheduling problem. In *Proc. 24th ACM Symposium on Theory of Computing (STOC)*, pages 51–58, 1992.

5 Y. Bartal, H. Karloff, and Y. Rabani. A better lower bound for on-line scheduling. *Inf. Process. Lett.*, 50(3):113–116, 1994.

6 B. Chen, A. van Vliet, and G. Woeginger. A lower bound for randomized on-line scheduling algorithms. *Inf. Process. Lett.*, 51(5):219–222, 1994. Publisher: Elsevier.

7 L. Chen, D. Ye, and G. Zhang. Approximating the optimal algorithm for online scheduling problems via dynamic programming. *Asia-Pacific Journal of Operational Research*, 32(01):1540011, 2015. Publisher: World Scientific.

8 T.C.E. Cheng, H. Kellerer, and V. Kotov. Semi-on-line multiprocessor scheduling with given total processing time. *Theoretical computer science*, 337(1-3):134–146, 2005. Publisher: Elsevier.

9 J. Dohrau. Online makespan scheduling with sublinear advice. In *41st International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM)*, pages 177–188. Springer, 2015.

10 E.B. Dynkin. The optimum choice of the instant for stopping a Markov process. *Soviet Mathematics*, 4:627–629, 1963.

11 M. Englert, D. Özmen, and M. Westermann. The power of reordering for online minimum makespan scheduling. In *Proc. 49th 676 IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 603–612. IEEE, 2008.

12 U. Faigle, W. Kern, and G. Turán. On the performance of on-line algorithms for partition problems. *Acta cybernetica*, 9(2):107–119, 1989.

13 M. Feldman, O. Svensson, and R. Zenklusen. A simple O (log log (rank))-competitive algorithm for the matroid secretary problem. In *Proc. 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1189–1201. SIAM, 2014.

14 R. Fleischer and M. Wahl. On-line scheduling revisited. *Journal of Scheduling*, 3(6):343–353, 2000. Publisher: Wiley Online Library.

15 G. Galambos and G. Woeginger. An on-line scheduling heuristic with better worst-case ratio than Graham's list scheduling. *SIAM Journal on Computing*, 22(2):349–355, 1993. Publisher: SIAM.

16 G. Goel and A. Mehta. Online budgeted matching in random input models with applications to Adwords. In *SODA*, volume 8, pages 982–991, 2008.

17 T. Gormley, N. Reingold, E. Torng, and J. Westbrook. Generating adversaries for request-answer games. In *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, pages 564–565, 2000.

18 R. Graham. Bounds for certain multiprocessing anomalies. *Bell System Technical Journal*, 45(9):1563–1581, 1966. Publisher: Wiley Online Library.

19 Anupam Gupta, Ruta Mehta, and Marco Molinaro. Maximizing Profit with Convex Costs in the Random-order Model. In *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*, volume 107, pages 71:1–71:14, 2018. doi:10.4230/LIPIcs.ICALP.2018.71.

20 D.S. Hochbaum and D.B. Shmoys. Using dual approximation algorithms for scheduling problems theoretical and practical results. *J. ACM*, 34(1):144–162, 1987. Publisher: ACM New York, NY, USA.

21 C. Karande, A. Mehta, and P. Tripathi. Online bipartite matching with unknown distributions. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 587–596, 2011.

22 D.R. Karger, S.J. Phillips, and E. Torng. A better algorithm for an ancient scheduling problem. *Journal of Algorithms*, 20(2):400–430, 1996. Publisher: Elsevier.

23 H. Kellerer and V. Kotov. An efficient algorithm for bin stretching. *Operations Research Letters*, 41(4):343–346, 2013. Publisher: Elsevier.

24 H. Kellerer, V. Kotov, M.G. Grazia Speranza, and Z. Tuza. Semi on-line algorithms for the partition problem. *Operations Research Letters*, 21(5):235–242, 1997. Publisher: Elsevier.

25 C. Kenyon. Best-Fit Bin-Packing with Random Order. In *SODA*, volume 96, pages 359–364, 1996.

26 T. Kesselheim, A. Tönnis, K. Radke, and B. Vöcking. Primal beats dual on online packing LPs in the random-order model. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 303–312, 2014.

27 R.D. Kleinberg. A multiple-choice secretary algorithm with applications to online auctions. In *SODA*, volume 5, pages 630–631, 2005.

28 O. Lachish. O (log log rank) competitive ratio for the matroid secretary problem. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 326–335. IEEE, 2014.

29 M. Mahdian and Q. Yan. Online bipartite matching with random arrivals: an approach based on strongly factor-revealing lps. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 597–606, 2011.

30 A. Meyerson. Online facility location. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 426–431. IEEE, 2001.

31 M. Molinaro. Online and random-order load balancing simultaneously. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1638–1650. SIAM, 2017.

32 C.J. Osborn and E. Torng. List's worst-average-case or WAC ratio. *Journal of Scheduling*, 11(3):213–215, 2008. Publisher: Springer.

33 K. Pruhs, J. Sgall, and E. Torng. *Online scheduling*. CRC Press, 2004.

34 J.F. Rudin III. Improved bounds for the on-line scheduling problem, 2001.

35 J.F. Rudin III and R. Chandrasekaran. Improved bounds for the online scheduling problem. *SIAM Journal on Computing*, 32(3):717–735, 2003. Publisher: SIAM.

36 P. Sanders, N. Sivadasan, and M. Skutella. Online scheduling with bounded migration. *Mathematics of Operations Research*, 34(2):481–498, 2009. Publisher: INFORMS.

37 J. Sgall. A lower bound for randomized on-line multiprocessor scheduling. *Inf. Process. Lett.*, 63(1):51–55, 1997. Publisher: Citeseer.

38 D.D. Sleator and R.E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985. Publisher: ACM New York, NY, USA.